Universidad
Carlos III de Madrid

Autor:
Diego García-Vaquero Fernández

Trabajo de Fin de Grado
2015-2016

Grado en Ingeniería Informática

TideCAD: EDITOR DE FICHEROS AUTOCAD

Tutor del proyecto:
Félix García Carballeira

Autor:
Diego García-Vaquero Fernández

Agradecimientos

Con la realización de este Trabajo Fin de Grado concluye otra etapa más de mi vida. Sin duda, una de las etapas más importantes en la que he conseguido ampliar mi formación académica y alcanzar la madurez como persona. Sin embargo, ha sido un proceso difícil que no hubiera podido culminar solo.

En primer lugar, me gustaría agradecer a mis padres el esfuerzo, tanto personal como económico, que han realizado y su fe ciega en mí. Soy lo que soy gracias a ellos y les estaré eternamente agradecidos.

A todos los profesores que me han impartido clase a lo largo de la carrera, en especial a los de las asignaturas de Programación, Estructuras de Datos, Interfaces de Usuario e Informática Gráfica. Gracias a sus enseñanzas he sabido sacar adelante este proyecto.

A mi tutor Félix, por su orientación y sus consejos durante todo el proyecto.

A todos los compañeros que he tenido durante la carrera y que ahora puedo llamar amigos. Por todos los momentos en los que el trabajo en grupo ha sido base para la consecución de los éxitos.

Por último, me gustaría agradecer a mi novia la paciencia que ha tenido conmigo en los momentos difíciles que he tenido que afrontar en estos cuatro años.

Índice general

Índice de ilustraciones

Índice de tablas

1. INTRODUCCIÓN

En las siguientes líneas se procede a realizar una breve introducción del proyecto, mencionando tanto los objetivos que se persiguen con la realización del mismo, como la motivación inicial para desarrollarlo. También se comentará brevemente la estructura del documento.

1.1. Motivación del proyecto

En este epígrafe, se hablará de sobre cuáles fueron los alicientes, tanto personales como externos, que se dieron para que el proyecto se llevara a cabo.

En primer lugar, y tras cursar la asignatura *Sistemas Distribuidos* con el tutor de este proyecto, Félix García Carballeira, acudí a él para que me comentara alguna idea sobre la que yo pudiera desarrollar mi Trabajo Fin de Grado (en adelante *TFG*). Él me comentó la existencia de una librería, *Kabeja*, que nos permite obtener información de dibujos de diseño asistido por computador en formato *Drawing Exchange Format* (en adelante *dxf*), así como poder convertir dichos archivos a otros formatos. La idea era realizar una interfaz gráfica sobre la que el usuario pudiera abrir y manipular diferentes ficheros en el formato mencionado con anterioridad utilizando Java como lenguaje de programación.

Java ha sido el lenguaje con el que, en primero, nos enseñaron a programar y si bien es cierto que, aunque debido a la especialidad escogida en las últimas asignaturas cursadas hemos programado más en lenguaje C y C++, el lenguaje con el que empiezas siempre es con el que te sientes más cómodo a la hora de programar. Sin embargo, mi nivel de Java es un nivel básico, puesto que lo aprendimos en los primeros años de la carrera y además es un lenguaje que ofrece muchas posibilidades y el dominarlo es una tarea complicada.

Uno de los aspectos que no pudimos ver en el desarrollo de los estudios fue el de las Interfaces Gráficas de Usuario (*GUI*) para este lenguaje de programación, porque aunque vimos los conceptos teóricos en la asignatura *Interfaces de Usuario*, el trabajo práctico lo dedicamos a desarrollo web. El tema de las *GUI's* en Java siempre me había causado gran curiosidad y lo tenía pendiente para cuando terminase el grado; por lo que fue un aliciente más para optar por realizar este proyecto.

En cuanto a la temática del proyecto, la idea de poder abrir y modificar un fichero en formato *dxf*, un formato para dibujos de diseño asistido por computador (en adelante *CAD*) y usado

por el programa *AutoCAD* y el resto de programas del mercado [1], me parecía un reto al que me apetecía enfrentarme y del que iba a aprender diversos aspectos nuevos y desconocidos para mí hasta ese momento. Por otro lado, *AutoCAD* es la principal herramienta utilizada en ingeniería para la realización de diseños, por lo que poder aprender más sobre ella y poder realizar una aplicación básica en la que pudiera obtener características de los ficheros generados por *AutoCAD* era una idea, cuanto menos, tentadora.

1.2. Objetivos

El principal objetivo del proyecto es el de desarrollar una aplicación de libre distribución capaz de abrir ficheros en formato *dxf*, mostrar su contenido y poder modificarlo añadiendo una serie de entidades que se describirán en apartados posteriores. Además, la aplicación podrá exportar el fichero, modificado o no, a una serie de formatos estándares como son: SVG, JPG, PNG, PDF y TIFF.

Para lograrlo, se procede en primer lugar al diseño e implementación de la interfaz con la que el usuario podrá interactuar con la aplicación. Después se implementará la funcionalidad de los distintos componentes de la misma para lograr el objetivo principal descrito en el párrafo anterior.

Además del objetivo más importante, surgen una serie de objetivos secundarios asociados al proyecto que se detallan a continuación:

- Comprender el funcionamiento de una *GUI* en Java y aprender a implementarla.

- Fusionar el código desarrollado para la aplicación con la funcionalidad proporcionada por *Kabeja*.

- Agregar a la aplicación la funcionalidad básica de un editor de ficheros: *zoom, pan, undo, redo,* entre otros.

1.3. Contenido de la memoria

A continuación se procede a describir cada una de las partes que comprenden el documento:

- Introducción: a modo de toma de contacto con el proyecto, se comentan cuáles han sido las motivaciones para la realización del mismo, los objetivos que se pretenden y el contenido del documento.

- Estado del Arte: se analizan las distintas tecnologías empleadas en la realización del proyecto junto con un estudio de las aplicaciones similares a la desarrollada en este proyecto.

- Análisis del Sistema: se obtiene una especificación detallada del sistema que se va a construir. Así mismo, servirá como base para el posterior diseño del sistema.

- Diseño del Sistema: consiste en resolver el problema que se ha descrito y modelado en el Análisis del Sistema. En este apartado se van a describir las decisiones tomadas para el diseño de la aplicación.

- Implementación y Pruebas: en este proceso se va a definir cómo se soluciona el problema planteado, detallando los algoritmos y técnicas concretas a utilizar. También se realiza una especificación detallada de las pruebas ejecutadas para la comprobación del correcto funcionamiento del sistema.

- Gestión del Proyecto: se realiza un estudio del tiempo empleado para el desarrollo del mismo y se compara con la planificación inicial propuesta.

- Conclusiones y trabajos futuros: se comentan cuáles han sido los conceptos aprendidos durante el desarrollo del proyecto y las posibles mejoras a tener en cuenta para una posterior mejora del mismo.

- Resumen en inglés: recapitulación del documento en inglés incluyendo los apartados de introducción, objetivos, resultados y conclusiones.

Además de los contenidos anteriormente citados, se añaden los siguientes anexos:

- ANEXO A: Acrónimos y Glosario de términos.
  Incluye acrónimos y siglas, y definiciones de los términos o expresiones que puedan tener interpretaciones ambiguas o no quedasen claras durante el desarrollo de la memoria del proyecto.

- ANEXO B: Referencias bibliográficas.
  Incluye referencias a la documentación utilizada para la realización del proyecto.

- ANEXO C: Manual de usuario.
  Explicación detallada del funcionamiento de la aplicación.

2. ESTADO DEL ARTE

En este apartado se realizará un estudio previo en el que se analizarán las herramientas similares a la implementada en este TFG y se comentarán en profundidad las distintas tecnologías utilizadas durante el desarrollo del mismo.

Como se ha comentado en la introducción, el lenguaje sobre el que se ha desarrollado el proyecto ha sido Java, utilizando como *framework* para su programación *Eclipse Kepler*. Además, se explicarán las librerías, tanto de Java como externas, que se han utilizado para el desarrollo de *TideCAD*.

2.1. Java

Java es un lenguaje de programación orientado a objetos que tuvo su gran auge cuando su primera versión comercial, la JDK 1.0, fue lanzada en 1996 por parte de Sun Microsystems. A día de hoy es uno de los lenguajes más usados para la programación en todo el mundo [2]. Se podría decir que Java es una evolución de los lenguajes C y C++ y es el lenguaje en el que ya se establece definitivamente el concepto de programación orientada a objetos.

La Programación Orientada a Objetos (en adelante POO) es una técnica de programación con la que podemos desarrollar soluciones computacionales utilizando componentes de software (objetos de software). En esta técnica, podemos distinguir los siguientes elementos: [3]

- Objeto: Componente de software que contiene tanto sus características (campos) como sus comportamientos (métodos); se accede a través de su interfaz.

- Campo: Característica de un objeto. Ayuda a definir su estructura y permite diferenciarlo de otros objetos. Se define con un identificador y un tipo, que nos indica los valores que puede almacenar. El estado del objeto queda definido por el conjunto de valores de los campos.

- Método: Implementación de un algoritmo que representa una operación o función que un objeto realiza. El comportamiento de un objeto lo determina el conjunto de métodos asociados a él.

2.1.1. Historia

La historia de Java comienza allá por los años 90, cuando James Gosling, ingeniero de *Sun Microsystems*, junto a varios compañeros, entre los que podemos destacar a Patrick Naughton y Mike Sheridan, comenzó a trabajar en *The Green Project*, un proyecto al margen de la compañía en la que trabajaban. El objetivo del mismo era desarrollar una nueva tecnología para la programación de los nuevos dispositivos inteligentes que estaban por venir. Además, buscaban que fuese un lenguaje fácil de aprender y de usar.

Su primera idea fue la de partir de C++ y modificarlo, puesto que podían reutilizar muchos aspectos de este lenguaje; sin embargo, finalmente decidieron por crear un nuevo lenguaje: Oak, que tenía similitudes con C, C++ y Objective C. Por motivos de propiedad intelectual, el nombre fue sustituido por Java.

En 1992, el equipo desarrolló un sistema inicial en un prototipo llamado *Star7,* y fue en este momento cuando el presidente de *Sun*, Scott McNealy, se percató del auge del proyecto y lo estableció como una subsidiaria de su compañía. El 3 de septiembre de este año terminaron el desarrollo y se puso fin al “Proyecto Verde”.

El cese de este proyecto coincidió con el nacimiento de la WEB, y sus miembros se dieron cuenta de que su proyecto se adecuaba muy bien a este nuevo ambiente; por lo que crearon un prototipo de navegador, *WebRunner*, que más tarde sería conocido como *HotJava*. Con el paso del tiempo, *HotJava* se convierte en un concepto bastante práctico dentro del lenguaje Java y demuestra, al proporcionar multiplataforma, que el código puede ser bajado del host del *World Wide Web* y ejecutado. También da soporte a los *applets*, que son componentes de aplicaciones que pueden ser ejecutados en el contexto de otro programa, como por ejemplo, un navegador web [4].

Es en 1995 cuando, en la conferencia *SunWorld,* se anuncia la versión *alpha* de Java y el acuerdo de *Sun* con *Netscape* para la incorporación de Java en los navegadores de esta última compañía que, en aquel momento, eran los más utilizados del mercado. Al año siguiente, se fundó *JavaSoft*, el grupo empresarial que se encargaría de los desarrollos tecnológicos, y se lanzó la primera versión de Java: JDK 1.0. En 2010, *Oracle* adquiere *Sun.* [5] [6]

La filosofía con la que *Gosling* comenzó el proyecto era la de: "Escríbelo una vez, ejecútalo en cualquier lugar", es decir, buscaba la posibilidad de desarrollar programas que pudieran ser ejecutados en un sinfín de plataformas sin la necesidad de recompilar [7].

2.1.2. Versiones

Entre 1996 y la actualidad, han sido varias las versiones de la plataforma que la compañía ha sacado al mercado. A lo largo de los años, han sido muchos los cambios y mejoras que se han realizado sobre la tecnología: [8] [9]

- JDK 1.0: primer lanzamiento.

- JDK 1.1: en esta segunda versión, se reestructura el modelo de eventos AWT y se añaden funcionalidades como *JavaBeans* o *JDBC* para integrar las bases de datos.

- J2SE 1.2: se añade reflexión a la programación, que consiste en la capacidad que tienen algunos programas en observar y modificar su estructura de alto nivel. Además, se incorpora la API gráfica (*Java Swing*) y las colecciones.

- J2SE 1.3: las incorporaciones más destacadas son las de *JavaSound, JNDI,* que permite a los usuarios el localizar objetos a través de un nombre y *JPDA*, para depurar aplicaciones.

- J2SE 1.4: en esta nueva versión se introducen las API para entrada y salida no bloqueante y para la lectura y escritura de imágenes en formatos JPEG y PNG. También integra un *parser XML* y *Java Web Start*, que permite arrancar aplicaciones Java localizadas en un servidor Web.

- J2SE 5.0: añade una gran cantidad de nuevas funcionalidades a la plataforma, entre las que podemos destacar las plantillas, los metadatos, las conversiones entre tipos primitivos y las enumeraciones.

- Java SE 6: incluye mejoras en el rendimiento y un cliente completo de Servicios Web, además de una serie de API's que hacen posible la combinación de Java con otros lenguajes como Python o Ruby.

- Java SE 7: amplía la plataforma con la introducción de anotaciones para detectar posibles fallos en el software y da soporte para *XML* y para *clausuras*, que son funciones examinadas en un entorno pero que tienen variables que dependen de otros.

- Java SE 8: incorpora por completo la librería *JavaFX* y añade diferentes mejoras en seguridad y concurrencia.

Como podemos observar en la lista anterior, las primeras versiones se lanzaban con las siglas *JDK (Java Development Kit)*. A partir de la versión 1.2, se sustituyeron las siglas anteriores por *J2SE (Java 2 Platform, Standard Edition)*. Esto se hizo para diferenciar a la versión base de las versiones empresarial (*J2EE*) y reducida (*J2ME*). Por último, en las versiones finales se elimina el ".0" y únicamente se usa el *SE* [10].

En la siguiente tabla, a modo de resumen, podemos observar la fecha de lanzamiento de las diferentes versiones y el nombre clave de cada una de ellas:

Version Name	Code Name	Release Date
JDK 1.0	Oak	January 1996
JDK 1.1	(none)	February 1997
J2SE 1.2	Playground	December 1998
J2SE 1.3	Kestrel	May 2000
J2SE 1.4	Merlin	February 2002
J2SE 5.0	Tiger	September 2004
Java SE 6	Mustang	December 2006
Java SE 7	Dolphin	July 2011
Java SE 8	*(Not available)*	March 2014

Ilustración 1: Nombre, nombre clave y fecha de lanzamiento de las distintas versiones de Java [11]

2.1.3. Arquitectura

Como se ha indicado en epígrafes anteriores, Java es un lenguaje multiplataforma, es decir, los paquetes de Java ya incluyen los recursos que necesita el sistema operativo para ejecutar el código java. El modo de que esto sea posible se debe a que Java tiene

un intérprete o máquina virtual (*JVM*) para cada una de las plataformas sobre las que se desea ejecutar su código. Sin embargo, además del código, necesita un conjunto de bibliotecas que contengan clases y objetos específicos de la plataforma. Esta combinación de máquina virtual y bibliotecas se conoce como Entorno de Ejecución de Java (*JRE*) [11].

Además del *JRE*¸ Java cuenta con un recolector de basuras que permite evitar el problema de *fugas de memoria* que sí sufren otros lenguajes. Básicamente, el recolector de basuras se encarga de liberar la memoria de los objetos creados cuando no quedan referencias a estos [12].

2.1.4. Java Swing

Java Swing es una biblioteca gráfica para Java que permite la implementación de GUI's. Como se ha visto en epígrafes anteriores, fue añadida en la versión J2SE 1.2 y está contenida en el paquete *javax.swing*. En este punto, es necesario comentar que todos los paquetes que cuelgan de la raíz *javax* son extensiones de Java [13].

Antes de la inclusión de esta biblioteca, Java contaba con otra biblioteca de componentes gráficos, *AWT* [14]; por lo que se puede decir que *Java Swing* fue creada a partir de esta [15].

Podemos ver su estructura en la siguiente imagen:

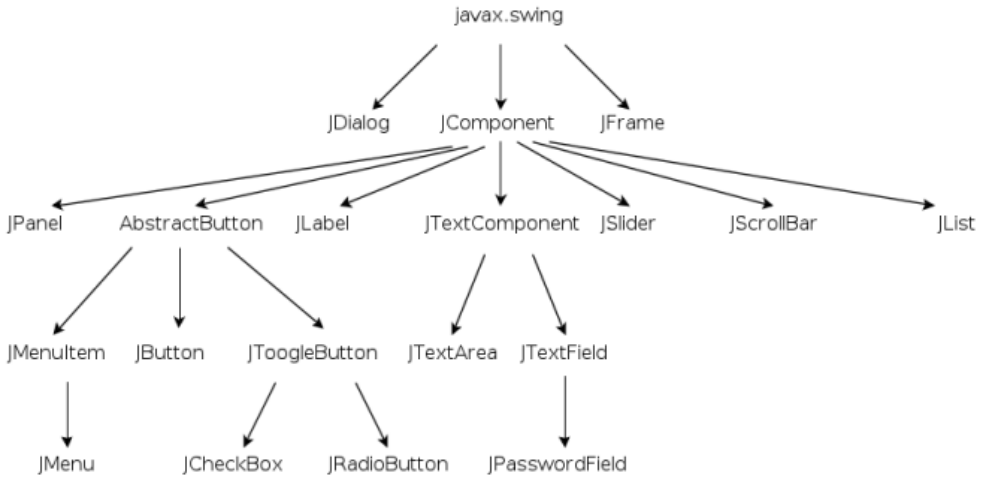

Ilustración 2: Estructura del paquete javax.swing [15]

Como podemos observar, del paquete raíz se derivan tres tipos de clases: *JDialog*, que nos permitirá construir los diálogos; *JComponent* del que heredarán el resto de componentes; y *JFrame*, que servirá como base para la aplicación principal y con la que se construyen las distintas ventanas.

El resto de componentes se corresponden con los *widgets* que podemos añadir a una interfaz gráfica de usuario: botones, menús, etiquetas o listas, entre otros.

2.1.5. Clase *Graphics*

La clase *Graphics* es la clase abstracta, base de todos los contextos gráficos, que permite a una aplicación dibujar en componentes desplegados en diversos dispositivos. Un objeto de esta clase encapsula toda la información necesaria para las operaciones de *renderización* que soporta Java.

Las coordenadas utilizadas son infinitamente delgadas y se encuentran en los píxeles de los dispositivos de salida. Además, todas las coordenadas que se utilizan en los métodos de un objeto de esta clase se trasladan al origen antes de hacer la llamada a los mismos.

Las operaciones se pueden dividir en tres grupos: [16]

- Operaciones que dibujan el contorno de las figuras establecen un camino infinitamente delgado entre los píxeles.
- Operaciones que rellenan una figura operan rellenando el interior de ese camino infinitamente delgado.
- Operaciones que representan texto horizontal se encargan de dibujar los glifos de caracteres tomando una línea base de coordenadas.

Los objetos de la clase *Graphics* no pueden ser instanciados directamente, por lo que se debe crear un componente y pasarlo como argumento del método *paint()*, que será el único tipo de argumento que soporte dicho método. La clase *Graphics* tiene métodos para soportar las tres categorías de operaciones gráficas:

- Dibujo de primitivas gráficas.
- Dibujo de texto.
- Presentación de imágenes en formato *.gif y *.jpeg.

Además de las operaciones anteriores, también mantiene un contexto grafico que comprende el área de dibujo actual, el color del fondo y del primer plano y un objeto *Font* con todas sus propiedades, entre otros. Por último, es necesario destacar que el origen de coordenadas está situado en la esquina superior izquierda [17].

2.2. *Eclipse*

Eclipse es una plataforma de código abierto basada en Java que permite a un desarrollador de software la creación de un entorno de desarrollo integrado (IDE, del inglés Integrated Development Environment) personalizado a partir de *plug-ins* creados por los miembros de Eclipse [18].

Fue desarrollada por IBM a finales de los años 90. El objetivo de la compañía era el de establecer una plataforma de desarrollo común para todos sus productos evitando el tener que reutilizar los elementos de la infraestructura que eran comunes. Además, al desarrollar *Eclipse* buscaban integrar todas las herramientas de IBM en una sola.

En noviembre de 1998, el grupo de software de IBM comenzó con *Eclipse*. Más específicamente fueron sus laboratorios OTI los que se encargaron del desarrollo del mismo; pero no fue hasta 2001 cuando la compañía decidió adoptar la licencia de código abierto para incrementar su adopción por parte de los desarrolladores de todo el planeta. Por tanto, la *open-source community* sería la propietaria del código; pero el derecho del marketing y las relaciones comerciales de la herramienta serían de la empresa.

En 2004, viendo los progresos que estaba teniendo la herramienta, nace la Fundación Eclipse: una entidad legal independiente de IBM y sin ánimo de lucro para ocuparse del desarrollo de Eclipse [19].

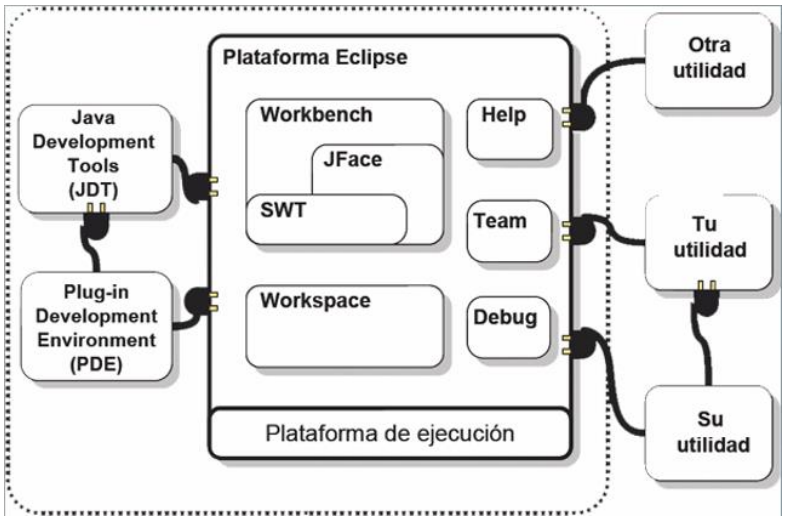

Ilustración 3: Arquitectura de Eclipse [20]

Como se ha comentado con anterioridad, *Eclipse* permite ser personalizado mediante la inclusión de *plug-ins*. Estos proporcionan extensibilidad a la aplicación, se ejecutan sobre el micro núcleo de *Eclipse* y son los que le verdaderamente de dan la funcionalidad al programa. El más utilizado es el *Java Development Tools* (en adelante, JDT), que permite el desarrollo de código java [20].

Además, *Eclipse* incorpora un Entorno de desarrollo de complementos (PDE, del inglés Plug-in Development Environment) que permite a los usuarios la posibilidad de crear sus propios *plug-ins.* Por otro lado, aunque esté basado en *Java*, la herramienta incluye soporte para la programación en otros lenguajes como C, C++ o COBOL.

En cuanto al resto de componentes de su arquitectura tenemos: [21]

- *Workbench* o entorno de trabajo: que proporciona a *Eclipse* una interfaz de usuario.
- *Standard Widget Toolkit (*en adelante, SWT*)*: que según la propia página de *Eclipse* "es un toolkit de código abierto para Java diseñado para proporcionar un acceso eficaz, portátil para las instalaciones de interfaz de usuario de los sistemas operativos en los que se aplique" [22].
- *JFace*: es un conjunto de componentes que permite realizar interfaces de usuario sobre SWT.
- *Workspace* o espacio de trabajo: se encarga de administrar los recursos del usuario. Incluye todos los proyectos del usuario y es el encargado de notificar al resto de componentes los cambios que se produzcan en los mismos.

- *Team* o soporte de equipo: es el encargado de proporcionar soporte para el control de la versión y la gestión de la configuración.
- *Help*: proporciona una estructura de navegación de *plug-ins* que permite a las herramientas agregar documentación en forma de archivos HTML.
- *Debug* o depurador: que permite realizar un seguimiento del código para encontrar posibles errores lógicos en el código.

Por último, en la siguiente tabla se muestran cuáles han sido las distintas versiones de Eclipse que se han lanzado y que versión de la plataforma traía cada una de ellas:

Nombre clave	Fecha de lanzamiento	Versión de la plataforma
Neon	Junio de 2016	4.6
Mars	24 de junio de 2015	4.5
Luna	25 de junio de 2014	4.4
Kepler	26 de junio de 2013	4.3
Juno	27 de junio de 2012	4.2
Indigo	22 de junio de 2011	3.7
Helios	23 de junio de 2010	3.6
Galileo	24 de junio de 2009	3.5
Ganymede	25 de junio de 2008	3.4
Europa	29 de junio de 2007	3.3
Callisto	30 de junio de 2006	3.2
Eclipse 3.1	28 de junio de 2005	3.1
Eclipse 3.0	28 de junio de 2004	3.0

Tabla 1: Nombre clave, fecha de lanzamiento y versión de la plataforma de las distintas versiones de Eclipse [23]

2.2.1. WindowBuilder Pro Eclipse

WindowBuilder Pro Eclipse es un *plug-in* de Eclipse que permite desarrollar una GUI por medio de una paleta de componentes mediante la técnica de *drag-and-drop*, generando el código necesario para su funcionamiento [24].

Para utilizarlo, es necesario añadir las librerías del componente en los archivos de *Eclipse*. Una vez instalado, el resultado lo podemos observar en la siguiente ilustración:

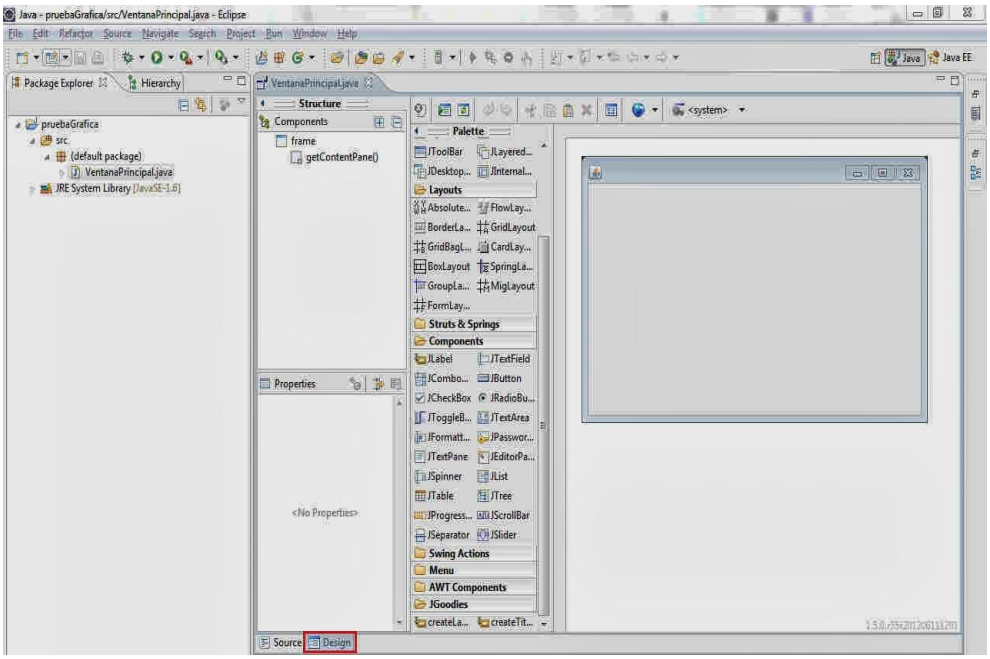

Ilustración 4: Imagen del plug-in WindowBuilder Pro Eclipse

2.3. Aplicaciones CAD

Hace 25 años, los diseños eran realizados con papel y lápiz, obligando a borrar o a volver a realizarlo si los cambios eran significativos. La incursión de los CAD modificó este método de trabajo; al principio con diseños en dos dimensiones (2D), que a medida de los años fueron evolucionando a las tres dimensiones (3D) y la realidad virtual.

Además del gran beneficio que supone el poder realizar un diseño en un ordenador en temas de rendimiento, calidad y productividad, paralelamente se han desarrollado otras herramientas relacionadas basadas en simulación, modelización y manufactura de productos, con las que se complementan a la perfección. Esta evolución se debe, en gran medida, al aumento de la capacidad de los procesadores y la facilidad de uso de los ordenadores [25].

2.3.1. Historia

El Diseño Asistido por Computador ha constituido un hito para el sector de la ingeniería, arquitectura y construcción; sobretodo, tal y como se ha indicado en la introducción de este apartado, porque eliminó la necesidad de realizar los diseños sobre papel. A

continuación, se comenta una breve historia del CAD, desde los años 50 en los que apareció el primer programa gráfico, hasta los años 90: [25]

- Antes de los años 70, los primeros contactos con CAD se realizaron en las Fuerzas Aéreas de los Estados Unidos de América, en colaboración con el Instituto Tecnológico de Massachusetts (MIT). Más tarde, se empezaron a utilizar las primeras aplicaciones CAD para diseñar espacios interiores de oficinas.

- En los años 70, varias compañías comenzaron a ofrecer sistemas de diseño automatizado. Es en esta época cuando tienen sus inicios los productos y firmas más conocidos en la actualidad, como CATIA o CADLink. A finales de esta década, un sistema CAD consistía en un ordenador pequeño de 16 bits con un máximo de 512kb de memoria y de 20 a 300 Mb de disco duro.

- En los años 80 hace su primera aparición *Autodesk* con el objetivo de crear una aplicación CAD que pueda funcionar sobre un PC. Este es el nacimiento de *AutoCAD*, que en poco tiempo se convirtió en una de los programas más populares de CAD y que sirvió de inspiración para otros programas de diversas compañías.

- Por último, en los años 90 se generalizan las visualizaciones 3D en estas aplicaciones. *AutoCAD* 12 se convierte en el programa CAD más vendido sobre *Windows* y mucha gente comienza a utilizarlos de forma habitual. Se desarrollan programas mejores para atraer la atención de las empresas y programas más sencillos para el usuario común.

2.3.2. AutoCAD

Como se ha comentado en el epígrafe anterior, *AutoCAD* es una aplicación software para dibujo en 2D y 3D. Fue desarrollado por *Autodesk* y lleva en el mercado desde 1982. El programa ha crecido sustancialmente con el paso de los años, ya que le han añadido nuevas características para atraer a usuarios de distintas profesiones. Además, en los últimos años también han apostado por desarrollarlo para aplicaciones móviles y aplicaciones basadas en la nube [26].

Los sectores donde el programa es más utilizado son la construcción y la animación: [26]

- En construcción, es sin duda la herramienta utilizada por excelencia. Combina modelado de información de construcción (BIM, Building Information Modeling) con herramientas CAD para ayudar a los ingenieros a diseñar, visualizar, simular y construir más eficientemente.

  Un ejemplo donde el uso del *AutoCAD* fue fundamental fue la construcción de la *Torre Shangai*. La torre era una estructura muy complicada y los diseñadores de pronto se dieron cuenta de que con las herramientas tradicionales eran incapaces de llevar a cabo el proyecto con éxito. Por tanto, tuvieron que utilizar el modelado BIM que les permitía obtener una previsualización del edificio y les ayudaba a tomar decisiones acertadas en cuanto a su diseño.

- La *Entertainment Creation suites* de *AutoCAD* permite producir gran cantidad de contenido animado en 3D, para que los animadores, artistas de efectos especiales y modeladores den rienda suelta a su creatividad, aumenten su productividad y cumplan con los plazos establecidos.

Desde su lanzamiento, han sido varias las versiones que se han desarrollado de la aplicación. En la siguiente tabla podemos ver la fecha de lanzamiento de cada versión así como su nombre clave:

Nombre oficial	Versión	Fecha de lanzamiento	Nombre clave
AutoCAD Versión 1.0	1	Noviembre de 1982	-
AutoCAD Versión 1.2	2	Abril de 1983	-
AutoCAD Versión 1.3	3	Septiembre de 1983	-
AutoCAD Versión 1.4	4	Noviembre de 1983	-
AutoCAD Versión 2.0	5	Octubre de 1984	-
AutoCAD Versión 2.1	6	Mayo de 1985	-
AutoCAD Versión 2.5	7	Junio de 1986	-
AutoCAD Versión 2.6	8	Abril de 1987	-
AutoCAD Versión R9	9	Septiembre de 1987	White Album
AutoCAD Versión R10	10	Octubre de 1988	Abbey Road
AutoCAD Versión R11	11	Octubre de 1990	Let it Be
AutoCAD Versión R12	12	Junio de 1992	-
AutoCAD Versión R13	13	Noviembre de 1994	-
AutoCAD Versión R14	14	Febrero de 1997	Sedona and Pinetop
AutoCAD 2000	15	Marzo de 1999	Tahoe
AutoCAD 2000i	16	Julio de 2000	Banff
AutoCAD 2002	17	Junio de 2001	Kirkland
AutoCAD 2004	18	Marzo de 2003	Reddeer
AutoCAD 2005	19	Marzo de 2004	Neo
AutoCAD 2006	20	Marzo de 2005	Rio
AutoCAD 2007	21	Marzo de 2006	Postrio
AutoCAD 2008	22	Marzo de 2007	Spago
AutoCAD 2009	23	Marzo de 2008	Raptor
AutoCAD 2010	24	Marzo de 2009	Gator
AutoCAD 2011	25	Marzo de 2010	Hammer
AutoCAD 2012	26	Marzo de 2011	Ironman
AutoCAD 2013	27	Marzo de 2012	Jaws
AutoCAD 2014	28	Marzo de 2013	Keystone
AutoCAD 2015	29	Marzo de 2014	Longbow
AutoCAD 2016	30	Marzo de 2015	Maestro
AutoCAD 2017	31	Marzo de 2016	Nautilus

Tabla 2: Nombre oficial, número de versión, fecha de lanzamiento y nombre clave de las distintas versiones de AutoCAD [27]

A lo largo de todas estas versiones, el programa ha mantenido una serie de características en común. Al igual que otras aplicaciones CAD, *AutoCAD* gestiona una base de datos de entidades geométricas con la que se puede operar a través de una pantalla gráfica en la que éstas aparecen y que se conoce como editor de dibujo. En las primeras versiones, la interacción se producía mediante una línea de comandos; pero en las versiones más modernas, se realiza a través de una GUI.

Además de lo anterior, *AutoCAD* procesa imágenes de tipo vectorial, aunque también admite ficheros de tipo fotográfico o mapa de bits. El programa permite organizar las entidades en capas o estratos, pudiendo añadirle características a estas como el color o el grafismo. La herramienta permite también la posibilidad de utilizar bloques para la reutilización de objetos.

Por otra parte, uno de los principales usos del programa es, como se ha mencionado con anterioridad, la realización de planos para la construcción; por lo que *AutoCAD* incorpora recursos tradicionales del grafismo en el papel, como el color, el grosor de las líneas y las texturas por tramos.

La extensión del fichero de *AutoCAD* es *DWG* aunque permite exportar también a otros formatos como *DXF*. Este último permite compartir ficheros de dibujo con otras plataformas CAD, ya que el formato *DWG* se lo reserva *AutoCAD* para sí mismo. Sin embargo, debido a la alta popularidad del programa, el formato *DWG* se ha convertido en el estándar de facto de los diseños asistidos por ordenador, por lo que se ha liberado y los nuevos programas CAD lo están empezando a incorporar [28].

Como glosario, se puede afirmar que *AutoCAD* permite a sus usuarios dibujar de una manera sencilla, ágil y rápida y evita las desventajas de los dibujos a mano. Sin embargo, la aplicación también tiene sus desventajas como que es una herramienta de pago, lo que supone una limitación para muchos usuarios. Además, *AutoCAD* es un programa complejo, que requiere de mucho estudio para conseguir dominarlo por completo [29].

2.3.3. Otras aplicaciones CAD

Aunque *AutoCAD* es la herramienta más utilizada para los diseños por ordenador, hay otro número de aplicaciones que también realizan la misma función y que además son gratuitas:

- LibreCAD: aplicación de código abierto enfocada al diseño en 2D de planos y esquemas. Es compatible con los estándares de diseño DXF y CXF y es muy práctica debido a que combina paneles para capas, comandos y librerías de recursos.

- DraftSight: es similar a LibreCAD. Ofrece soporte para los formatos más utilizados en CAD (DWG, DXF y DWT) y tiene una interfaz más limpia que el anterior. También permite la elaboración de formas complejas.

- FreeCAD: es una herramienta utilizada para el diseño CAD en 3D. Está más pulida que las anteriores en cuanto a la interfaz e incluye funciones básicas, como renderizado, testeado y reparación de código incorrecto.

- BRL-CAD: es una herramienta bastante antigua que está más enfocado al uso de comandos que al uso herramientas de dibujo.

- OpenSCAD: permite la creación de modelos 3D CAD que traduce a elementos gráficos a partir de scripts.

2.4. Bibliotecas de libre distribución para la manipulación de ficheros CAD

En este apartado se describen las principales bibliotecas de código abierto disponibles para el tratamiento de ficheros CAD.

2.4.1. *Kabeja*

Kabeja es una biblioteca Java para analizar, procesar y convertir ficheros en formato DXF. Puede ser utilizada por línea de comandos o incrustada en otra aplicación. Todos los datos analizados son accesibles con la API basada en el *Document Object Model* (DOM). Además, soporta la API de SAX, que permite el uso de XML en Java.

El sistema permite procesar, filtrar y convertir dibujos DXF a diferentes formatos de salida como SVG, JPEG, PNG, TIFF, PDF o XML [30].

Uno donde los sitios donde *Kabeja* es usada es en *Inkscape* para importar ficheros DXF. *Inkscape* es un editor gráfico de vectores que permite crear y editar gráficos vectoriales. Además, es gratuito y de código libre [31].

La versión más reciente de *Kabeja* es la 0.4. Hasta ésta, han sido varias las versiones y las modificaciones que se han efectuado sobre las librerías: [32]

- La primera versión de *Kabeja* fue lanzada el 15 de abril del año 2005. Dicha versión era actualizada prácticamente cada mes debido a la gran cantidad de *bugs* que los desarrolladores iban descubriendo.

- El 17 de junio de 2006 lanzan *Kabeja 0.2.* En esta versión ya se ha aceptado la licencia Apache Software License 2.0 debido a que ahora *Kabeja* está alojada en sourceforge.net.

- El 7 de diciembre de 2006 se lanza *Kabeja 0.3*. Entre las mejoras más destacadas están la extensión para *Inkscape*, el soporte para sombreado o *hatch* y nuevos filtros por capa y por dimensiones.

- La última versión, *Kabeja 0.4*, fue lanzada en marzo de 2008. Las mejoras de esta última versión fueron el soporte para nuevas entidades como *MLine y Spline*, la inclusión de una GUI y la separación del núcleo de procesamiento de la parte de generación de SVG.

Como ya se ha mencionado, el equipo de *Kabeja*, además de la librería que nos permite tratar con ficheros DXF, en la última versión ha desarrollado una aplicación en la que, además de implementar toda la funcionalidad de su librería, permite visualizarlos.

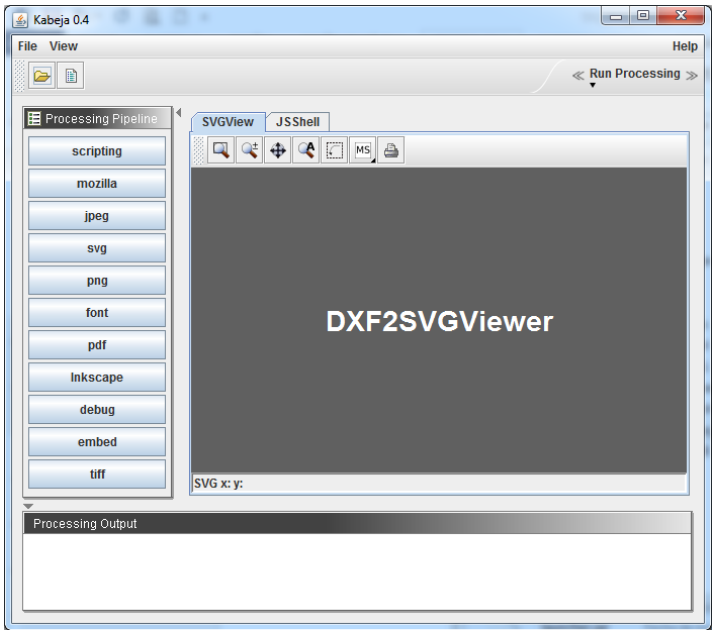

Ilustración 5: Aplicación de Kabeja

2.4.2. YCad

YCad es otra biblioteca que permite la manipulación de ficheros CAD. Al igual que *Kabeja*, en la actualidad solo soporta el formato DXF para leer, visualizar y escribir en este tipo de ficheros [33].

Para poder manipular la librería, es necesario crear una instancia de la clase Yxxf contenida en el paquete com.ysystems.ycad.lib.yxxf. Esta clase representa una instancia simple de un dibujo YCad. El objeto de dibujo Yxxf es una estructura de datos jerárquica basada en el formato DXF de Autodesk. El objeto de dibujo Yxxf contiene referencias a las distintas secciones DXF y *flags* que describen su estado. Es necesaria una relación con el formato DXF para comprender y utilizar el objeto Yxxf totalmente. Las secciones del documento DXF que son consultadas y necesarias para que el dibujo YCad pueda ser presentado son: [34]

- Header: una colección de campos individuales como el tipo de línea actual, color actual, la escala del tipo de línea.
- Tables: una colección de tablas que incluye: tabla de capas, tabla de tipo de línea, estilo del texto (fuentes), tabla Vport
- Entities: una secuencia de entidades de dibujo tales como línea, círculo, polilínea, el bloque de inserción. Hay dos tipos de entidades: las de “Espacio Papel” y “Espacio Modelo”.
- Blocks: una secuencia de entidades del dibujo identificadas con un nombre.

Al igual que *Kabeja*, *YCad* también cuenta con un applet para leer y representar el dibujo: Ycadv.

3. ANÁLISIS DEL SISTEMA

En este apartado se realizará un análisis exhaustivo del sistema desarrollado. En este proyecto y tal como se ha indicado en apartados anteriores, el sistema desarrollado es un prototipo de aplicación Java que nos permite la edición de ficheros de dibujo con formato DXF.

A continuación se detallarán los requisitos necesarios para el cumplimiento del sistema. Los requisitos se han subdividido en dos categorías principales: requisitos de usuario y requisitos de software. Además se comenta si es necesario tener en cuenta alguna restricción a la hora de realizar el proyecto. Al final de este apartado también se expondrán los diferentes casos de uso que puede manifestar la aplicación:

3.1. Requisitos de Usuario

Los Requisitos de Usuario (en adelante RU) están relacionados con una descripción de las funcionalidades que el sistema incluye. Tras el planteamiento inicial del problema, se procede a certificar la funcionalidad del sistema y el alcance del mismo.

Por las condiciones en las que se ha desarrollado el proyecto, la toma de requisitos se ha ido realizando a lo largo del desarrollo del mismo en varias sesiones de análisis con el tutor de este proyecto. Lo idóneo hubiera sido realizar una sesión inicial en la que se hubiera detallado una lista prácticamente definitiva de requisitos y sobre ella basar el diseño y posterior implementación.

Los RU se pueden clasificar en requisitos de capacidad, requisitos de restricción y requisitos inversos:

- Los requisitos de capacidad describen las funciones y operaciones que necesita el usuario para resolver un problema o alcanzar un objetivo.

- Los requisitos de restricción especifican las restricciones impuestas por el usuario de cómo el software se debe construir y cómo se deberá operar con él, es decir, restricciones de cómo se debe resolver el problema o cómo se debe alcanzar el objetivo.

- Los requisitos inversos se encargan de describir lo que el sistema no hará. Útiles especialmente para evitar confusiones.

Se detalla a continuación la plantilla que se utilizará para realizar la especificación de requisitos y se describen cada una de las partes de la misma:

Identificador: RUC/RUR/RUI-XX			
Prioridad	Alta / Media / Baja	**Fuente**	Tutor / Alumno / *Kabeja*
Necesidad	Esencial / Deseable / Opcional		
Claridad	Alta / Media / Baja	**Verificabilidad**	Alta / Media / Baja
Estabilidad			
Descripción			

Tabla 3: Plantilla de Requisitos de Usuario

A continuación se detallan cada uno de los elementos de la tabla anterior:

- Identificador: un código identificativo de cada requisito. Es conveniente para facilitar la trazabilidad. Está compuesto por un código alfanumérico que indica si el requisito es de capacidad (RUC), de restricción (RUR) o inverso (RUI) seguido del número identificativo dentro de su categoría.

- Prioridad: cada requisito incluye una medida de prioridad (alta, media o baja) para que el desarrollador pueda decidir la planificación del desarrollo.

- Fuente: se indica el origen de cada requisito de usuario. En el actual proyecto, los posibles orígenes son el tutor, el alumno o *Kabeja*, en el caso de que la librería imponga limitaciones.

- Necesidad: los requisitos esenciales del usuario son no negociables; el resto pueden ser menos importantes y sujetos a la negociación.

- Claridad: alta, media o baja según la claridad del requisito. Un requisito de usuario es claro si tiene una, y sólo una, interpretación, por lo que si hay múltiples interpretaciones será baja, si hay dos, media, y si hay solo una, alta.

- Verificabilidad: cada requisito de usuario será verificable. Esto significa que debe ser posible comprobar que el requisito se ha incorporado en el diseño, es decir, que se puede probar que el software aplica el requisito. En función del nivel en el que podamos comprobarlo, se puede clasificar en alta, media y baja.

- Estabilidad: algunos requisitos de usuario pueden permanecer estables durante toda la vida esperada del software; otros pueden ser más dependientes del diseño.

- Descripción: descripción detallada del requisito.

3.1.1. Especificación de RU de Capacidad

A continuación se detallan los requisitos que se corresponden con las necesidades de los usuarios:

Identificador: RUC-01			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario tiene la posibilidad de abrir y visualizar ficheros en formato DXF.		

Tabla 4: RUC-01: Abrir ficheros en formato DXF.

Identificador: RUC-02			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario tiene la posibilidad de modificar los ficheros previamente abiertos.		

Tabla 5: RUC-02: Modificar los ficheros abiertos

Identificador: RUC-03			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede salvar los cambios realizados en el fichero previamente abierto.		

Tabla 6: RUC-03: Salvar los cambios

Identificador: RUC-04			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede salvar los cambios realizados en el fichero previamente abierto en un nuevo fichero sin la modificación del fichero original abierto.		

Tabla 7: RUC-04: Salvar los cambios sin modificar el fichero original

Identificador: RUC-05			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede exportar el fichero previamente abierto a otros formatos.		

Tabla 8: RUC-05: Exportar a otros formatos

Identificador: RUC-06			
Prioridad	Media	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede elegir el color de las entidades con las que modifica los ficheros.		

Tabla 9: RUC-06: Elegir el color de las entidades

Identificador: RUC-07			
Prioridad	Media	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede elegir el grosor de las líneas que forman las entidades, salvo las de tipo “Texto”, con las que modifica los ficheros.		

Tabla 10: RUC-07: Elegir el grosor de las entidades

Identificador: RUC-08			
Prioridad	Media	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede elegir la continuidad de las líneas que forman las entidades, salvo las de tipo “Texto”, con las que modifica los ficheros.		

Tabla 11: RUC-08: Elegir la continuidad de las entidades

Identificador: RUC-09			
Prioridad	Media	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede elegir el tipo, tamaño y forma de las entidades de tipo “Texto”.		

Tabla 12: RUC-09: Elegir tipo, tamaño y forma de los textos

Identificador: RUC-10			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede ampliar o reducir el dibujo.		

Tabla 13: RUC-10: Ampliar o reducir el dibujo

Identificador: RUC-11			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede desplazarse por el dibujo tanto vertical como horizontalmente.		

Tabla 14: RUC-11: Desplazarse por el dibujo

Identificador: RUC-12			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede ajustar a la ventana el dibujo.		

Tabla 15: RUC-12: Ajustar a la ventana

Identificador: RUC-13			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede deshacer la última acción hecha.		

Tabla 16: RUC-13: Deshacer la última acción hecha

Identificador: RUC-14			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario puede rehacer la última acción deshecha.		

Tabla 17: RUC-14: Rehacer la última acción deshecha

3.1.2. Especificación de RU de Restricción

A continuación se detallan los requisitos que se corresponden con las restricciones impuestas por los usuarios:

Identificador: RUR-01			
Prioridad	Alta	**Fuente**	*Kabeja*
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El formato que los ficheros que puede abrir y editar el usuario es, únicamente, DXF.		

Tabla 18: RUR-01: Formato de archivos que puede abrir el usuario.

Identificador: RUR-02			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Susceptible a modificaciones.		
Descripción	Las entidades con las que el usuario puede modificar los ficheros abiertos son exclusivamente: • Línea • Polilínea • Rectángulo • Círculo • Arco • Texto		

Tabla 19: RUR-02: Entidades que puede añadir el usuario.

Identificador: RUR-03			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Susceptible a modificaciones.		
Descripción	Las entidades de las que el usuario obtiene información de los ficheros abiertos son las mismas que las especificadas en RUR-02.		

Tabla 20: RUR-03: Entidades que puede leer el usuario.

Identificador: RUR-04			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	Antes de poder añadir una entidad, el usuario debe ajustar a la pantalla el dibujo.		

Tabla 21: RUR-04: Restricción necesaria para añadir entidades.

Identificador: RUR-05			
Prioridad	Alta	**Fuente**	*Kabeja*
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	Los formatos a los que el usuario puede exportar el fichero abierto y/o modificado son: • SVG • JPG • PNG • PDF • TIFF		

Tabla 22: RUR-05: Formato de archivos que puede exportar el usuario.

Identificador: RUR-06			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	Los colores que el usuario puede asignarle a las entidades que añade a los ficheros son: • Rojo • Amarillo • Verde • Cyan • Azul • Magenta • Blanco • Gris		

Tabla 23: RUR-06: Tipos de colores para las entidades.

Identificador: RUR-07			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Susceptible a modificaciones.		
Descripción	Los valores que el usuario puede asignar a la continuidad de las líneas que forman las entidades son: • Continuo • Discontinuo		

Tabla 24: RUR-07: Tipos de continuidad para las entidades.

Identificador: RUR-08			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Susceptible a modificaciones.		
Descripción	Los valores que el usuario puede asignar al grosor de las líneas que forman las entidades son cuatro		

Tabla 25: RUR-08: Tipos de grosor para las entidades.

Identificador: RUR-09			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El número máximo de niveles de zoom son dos con respecto al nivel de zoom inicial, tanto para alejar como para acercar.		

Tabla 26: RUR-09: Niveles máximos de zoom.

Identificador: RUR-10			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	Los arcos tendrán un ángulo máximo de 180 grados.		

Tabla 27: RUR-10: Ángulo máximo de los arcos.

3.1.3. Especificación de RU Inversos

A continuación se detallan los requisitos que se corresponden con las acciones que los usuarios no pueden realizar en el sistema:

Identificador: RUI-01			
Prioridad	Alta	**Fuente**	*Kabeja*
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El usuario no puede crear un fichero de dibujo desde cero.		

Tabla 28: RUI-01: El usuario no puede crear un fichero de dibujo.

3.2. Marco regulador

Tras estudiar los requisitos y el marco vigente de legislación, se ha comprobado que no aplica ninguna ley sobre el proyecto, y que por tanto, no existe ninguna restricción sobre la realización del mismo.

3.3. Requisitos de Software

El propósito de la definición de Requisitos de Software (en adelante RS) es analizar los requisitos de usuario y producir un conjunto de requisitos que debe cumplir el software, tan completo, constante y correcto como sea posible. Esta especificación es tarea del desarrollador y debe establecerla de tal manera que satisfaga las necesidades de todos los interesados en el sistema.

El conjunto de RS, a su vez, se puede dividir en requisitos funcionales, no funcionales y negativos:

- Los requisitos funcionales pretenden definir el comportamiento interno del software y otras funcionalidades del sistema o de sus componentes, es decir, especifica el propósito del software. Se derivan de los requisitos de capacidad.

- Los requisitos no funcionales, en cambio, se encargan de restringir el modo de realizar las tareas, es decir, se centran más en el diseño y la implementación. Derivados de los requisitos de restricción.

- Los requisitos negativos especifican lo que la aplicación no debe o no necesita hacer. Se derivan de los requisitos inversos.

La plantilla que se utilizará para la especificación es la misma que la empleada para los RU (*Tabla 3*). El único cambio se establece en el código alfanumérico del identificador, que ahora empleará las siglas RSF, RSNF y RSN para referirse a los RS funcionales, a los RS no funcionales y a los RS negativos respectivamente.

3.3.1. Especificación de RS Funcionales

A continuación se detallan los requisitos que expresan la funcionalidad del sistema:

Identificador: RSF-01			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite abrir y visualizar ficheros en DXF.		

Tabla 29: RSF-01: Abrir ficheros en formato DXF.

Identificador: RSF-02			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite modificar ficheros previamente abiertos.		

Tabla 30: RSF-02: Modificar los ficheros abiertos.

Identificador: RSF-03			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite salvar los cambios realizados en el fichero previamente abierto.		

Tabla 31: RSF-03: Salvar los cambios.

Identificador: RSF-04			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación salvará los cambios realizados en el fichero previamente abierto en un nuevo fichero sin la modificación del fichero original abierto.		

Tabla 32: RSF-04: Salvar los cambios sin modificar el fichero original.

Identificador: RSF-05			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite exportar el fichero previamente abierto a otros formatos.		

Tabla 33: RSF-05: Exportar a otros formatos.

Identificador: RSF-06			
Prioridad	Media	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite al usuario elegir el color de las entidades con las que modifica los ficheros.		

Tabla 34: RSF-06: Elegir el color de las entidades.

Identificador: RSF-07			
Prioridad	Media	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite al usuario elegir el grosor de las líneas que forman las entidades, salvo las de tipo “Texto”, con las que modifica los ficheros.		

Tabla 35: RSF-07: Elegir el grosor de las entidades.

Identificador: RSF-08			
Prioridad	Media	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite al usuario elegir la continuidad de las líneas que forman las entidades, salvo las de tipo “Texto”, con las que modifica los ficheros.		

Tabla 36: RSF-08: Elegir la continuidad de las entidades.

Identificador: RSF-09			
Prioridad	Media	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite al usuario elegir el tipo, tamaño y forma de las entidades de tipo “Texto”.		

Tabla 37: RSF-09: Elegir tipo, tamaño y forma de los textos.

Identificador: RSF-10			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite ampliar o reducir el dibujo.		

Tabla 38: RSF-10: Ampliar o reducir el dibujo.

Identificador: RSF-11			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite al usuario desplazarse por el dibujo tanto vertical como horizontalmente.		

Tabla 39: RSF-11: Desplazarse por el dibujo.

Identificador: RSF-12			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite ajustar a la ventana el dibujo.		

Tabla 40: RSF-12: Ajustar a la ventana.

Identificador: RSF-13			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite deshacer la última acción hecha.		

Tabla 41: RSF-13: Deshacer la última acción hecha.

Identificador: RSF-14			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación permite rehacer la última acción deshecha.		

Tabla 42: RSF-14: Rehacer la última acción deshecha.

3.3.2. Especificación de RS No Funcionales

A continuación se detallan los requisitos que se corresponden con aspectos más relacionados con el *cómo* se debe desarrollar la aplicación antes que con el *qué*:

Identificador: RSNF-01			
Prioridad	Alta	**Fuente**	*Kabeja*
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El formato que los ficheros que puede abrir y editar la aplicación es, únicamente, DXF.		

Tabla 43: RSNF-01: Formato de archivos que puede abrir el usuario.

Identificador: RSNF-02			
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Susceptible a modificaciones.		
Descripción	Las entidades que la aplicación permite añadir a los ficheros abiertos son exclusivamente: • Línea • Polilínea • Rectángulo • Círculo • Arco • Texto		

Tabla 44: RSNF-02: Entidades que puede añadir el usuario.

Identificador: RSNF-03			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Susceptible a modificaciones.		
Descripción	Las entidades de las que la aplicación obtiene información de los ficheros abiertos son las mismas que las especificadas en RSNF-02.		

Tabla 45: RSNF-03: Entidades que puede leer el usuario.

Identificador: RSNF-04			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	Antes de añadir una entidad, la aplicación obliga al usuario a ajustar el dibujo a la pantalla.		

Tabla 46: RSNF-04: Restricción necesaria para añadir entidades.

Identificador: RSNF-05			
Prioridad	Alta	**Fuente**	*Kabeja*
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	Los formatos a los que la aplicación permite exportar el fichero abierto y/o modificado son: • SVG • JPG • PNG • PDF • TIFF		

Tabla 47: RSNF-05: Formato de archivos que puede exportar el usuario.

Identificador: RSNF-06			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	Los colores que la aplicación permite asignar a las entidades que añade a los ficheros son: • Rojo • Amarillo • Verde • Cyan • Azul • Magenta • Blanco • Gris		

Tabla 48: RSNF-06: Tipos de colores para las entidades.

Identificador: RSNF-07			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Susceptible a modificaciones.		
Descripción	Los valores que la aplicación permite asignar a la continuidad de las líneas que forman las entidades son: • Continuo • Discontinuo		

Tabla 49: RSNF-07: Tipos de continuidad para las entidades.

Identificador: RSNF-08			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Susceptible a modificaciones.		
Descripción	Los valores que la aplicación permite asignar al grosor de las líneas que forman las entidades son cuatro.		

Tabla 50: RSNF-08: Tipos de grosor para las entidades.

Identificador: RSNF-09			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	El número máximo de niveles de zoom son dos con respecto al nivel de zoom inicial, tanto para alejar como para acercar.		

Tabla 51: RSNF-09: Niveles máximos de zoom.

Identificador: RSNF-10			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La versión del JRE de Java debe ser la 1.7		

Tabla 52: RSNF-10: Versión de JRE de Java.

Identificador: RSNF-11			
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	Los arcos tendrán un ángulo máximo de 180 grados.		

Tabla 53: RSNF-11: Ángulo máximo de los arcos

3.3.3. Especificación de RS Negativos

A continuación se detallan los requisitos que se corresponden con las acciones que la aplicación no puede realizar en el sistema:

Identificador: RSN-01			
Prioridad	Alta	**Fuente**	*Kabeja*
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Durante toda la vida del sistema.		
Descripción	La aplicación no permite crear un nuevo fichero de dibujo.		

Tabla 54: RSN-01: El usuario no puede crear un fichero de dibujo.

3.4. Matriz de trazabilidad RU → RS

En el siguiente subapartado se va a realizar una matriz de trazabilidad entre los RU y los RS para comprobar que todo RU es desarrollado por al menos un RS. Pueden existir RS sin su correspondiente RU; pero no es muy común y debe existir una muy buena razón para que esto ocurra puesto que un cliente no querrá pagar por algo que no ha solicitado. En este proyecto el único RS de no tiene correspondencia con un RU es el que hace referencia a la versión *JRE* de Java.

La matriz de trazabilidad se muestra en la siguiente página:

	RSF-01	RSF-02	RSF-03	RSF-04	RSF-05	RSF-06	RSF-07	RSF-08	RSF-09	RSF-10	RSF-11	RSF-12	RSF-13	RSF-14	RSNF-01	RSNF-02	RSNF-03	RSNF-04	RSNF-05	RSNF-06	RSNF-07	RSNF-08	RSNF-09	RSNF-10	RSNF-11	RSN-01
RUC-01	X																									
RUC-02		X																								
RUC-03			X																							
RUC-04				X																						
RUC-05					X																					
RUC-06						X																				
RUC-07							X																			
RUC-08								X																		
RUC-09									X																	
RUC-10										X																
RUC-11											X															
RUC-12												X														
RUC-13													X													
RUC-14														X												
RUR-01															X											
RUR-02																X										
RUR-03																	X									
RUR-04																		X								
RUR-05																			X							
RUR-06																				X						
RUR-07																					X					
RUR-08																						X				
RUR-09																							X			
RUR-10																									X	
RUI-01																										X

Tabla 55: Matriz de trazabilidad RU -> RS

3.5. Casos de uso

En este apartado se mostrarán los casos de uso como una manera de acercar al lector de este TFG las funcionalidades de la aplicación *TideCAD*, sin profundizar en demasía y con el único objetivo de facilitar la comprensión de los aspectos más significativos de la misma.

En primer lugar, es necesario comentar que la técnica de los casos de uso permite también el poder capturar y definir los requisitos que debe cumplir una aplicación, y describe las típicas interacciones que hay entre el usuario y ésta.

En este proyecto el número de casos de uso dependerá del número de salidas (*Guardar, Guardar Como* y *Exportar*) que le podemos dar a nuestro fichero. Tendremos por tanto tres casos de uso.

Para la descripción de los casos de uso se utilizará la siguiente tabla:

Identificador: CU-XX	
Caso de Uso	
Actor	
Objetivo	
Precondiciones	
Postcondiciones	

Tabla 56: Plantilla de Casos de Uso

A continuación se procede a explicar cada uno de los campos de la tabla anterior que definen un caso de uso:

- Identificador: un código identificativo de cada caso de uso. Está compuesto por un código alfanumérico que indica que nos encontramos ante un caso de uso (CU), seguido de un número identificativo de cada uno de ellos.

- Caso de Uso: breve descripción del caso de uso.

- Actor: agente externo que interactúa con el sistema. En este caso, será siempre el usuario el encargado de dicha interactuación.

- Objetivo: finalidad del caso de uso o meta que se pretende alcanzar mediante la funcionalidad que describe.
- Precondiciones: se corresponde con el estado en el que se encuentra el sistema antes de la realización del caso de uso.
- Postcondiciones: estado que se alcanza después de la ejecución del caso de uso.

Además de la tabla, se adjuntará en cada caso de uso un diagrama para facilitar aún más la comprensión del mismo.

3.5.1. Caso de Uso 1: CU-01

Identificador: CU-01	
Caso de Uso	Caso de uso en el que, tras abrir el fichero, se amplía, desplaza y ajusta a la ventana el dibujo para después editarlo con una de las posibles entidades especificadas en los RU y guardarlo.
Actor	Usuario
Objetivo	Probar varias de las funcionalidades de la aplicación y después guardar el fichero.
Precondiciones	El usuario ha abierto un fichero y este se muestra en la aplicación.
Postcondiciones	El fichero abierto queda modificado según lo especificado en el caso de uso.

Tabla 57: CU-01: Caso de Uso 1

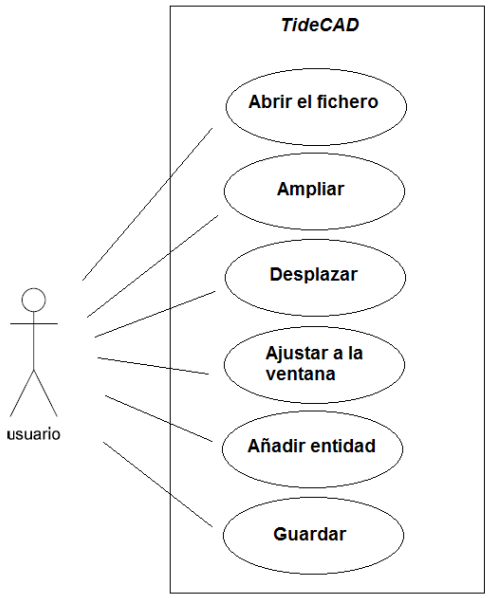

Ilustración 6: Diagrama CU-01

3.5.2. Caso de Uso 2: CU-02

Identificador: CU-02	
Caso de Uso	Caso de uso en el que, tras abrir el fichero, se edita con una de las posibles entidades especificadas en los RU modificando su color, continuidad y grosor. Después se guarda en un fichero diferente al original.
Actor	Usuario
Objetivo	Probar varias de las funcionalidades de la aplicación y después guardar en un fichero diferente al original.
Precondiciones	El usuario ha abierto un fichero y este se muestra en la aplicación.
Postcondiciones	El fichero abierto se mantiene inalterado mientras que se crea uno nuevo con el contenido modificado según lo especificado en el caso de uso.

Tabla 58: CU-02: Caso de Uso 2

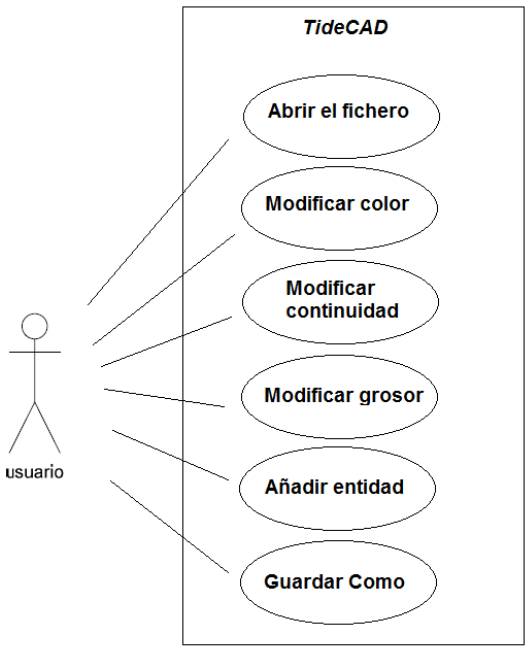

Ilustración 7: Diagrama CU-02

3.5.3. Caso de Uso 3: CU-03

Identificador: CU-03	
Caso de Uso	Caso de uso en el que, tras abrir el fichero, se edita con una entidad de tipo “Texto” eligiendo el tipo, tamaño y forma de la fuente. Después se deshace esta opción y se vuelve a rehacer y por último se exporta a uno de los formatos especificados en los RU.
Actor	Usuario
Objetivo	Probar varias de las funcionalidades de la aplicación y después exportar el fichero en formato diferente al original.
Precondiciones	El usuario ha abierto un fichero y este se muestra en la aplicación.
Postcondiciones	Se crea un nuevo fichero con el formato especificado en el caso de uso y con el contenido modificado.

Tabla 59: CU-03: Caso de Uso 3

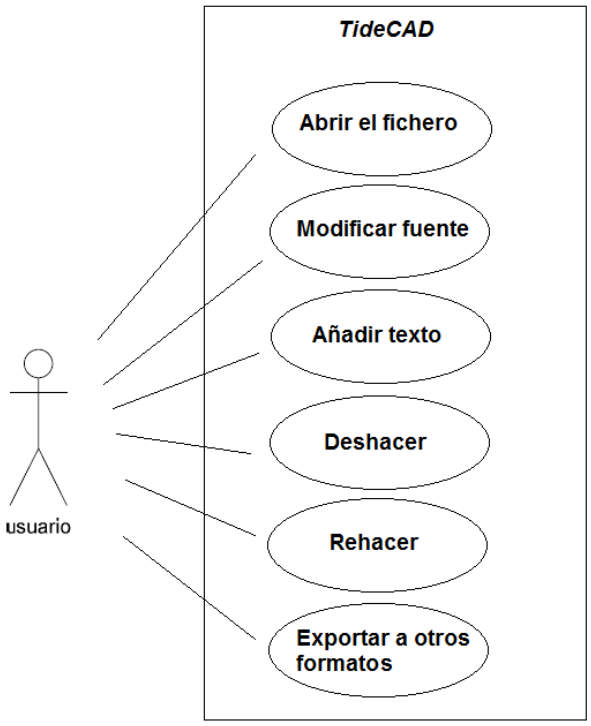

Ilustración 8: Diagrama CU-03

3.6. Matriz de trazabilidad RU → CU

En el siguiente subapartado se va a realizar una matriz de trazabilidad entre los RU y los CU para comprobar que todo RU aparece en al menos un CU, con la salvedad de los inversos que, por la filosofía de estos, no deben aparecer.

La matriz de trazabilidad se muestra en la siguiente página:

	CU-01	CU-02	CU-03
RUC-01	X	X	X
RUC-02	X	X	X
RUC-03	X		
RUC-04		X	
RUC-05			X
RUC-06		X	
RUC-07		X	
RUC-08		X	
RUC-09			X
RUC-10	X		
RUC-11	X		
RUC-12	X		
RUC-13			X
RUC-14			X
RUR-01	X	X	X
RUR-02	X	X	X
RUR-03	X	X	X
RUR-04	X		
RUR-05			X
RUR-06		X	
RUR-07		X	
RUR-08		X	
RUR-09	X		
RUR-10	X	X	X
RUI-01			

Tabla 60: Matriz de trazabilidad RU->CU

4. DISEÑO DEL SISTEMA

En este apartado se procede a comentar todas las decisiones de diseño que se han considerado a la hora de realizar el proyecto. Se hablará también de las posibles alternativas que podían haber sido escogidas para la realización del proyecto. A parte de la elección de Java como lenguaje sobre el que implementar la aplicación y *Kabeja* para la manipulación de los ficheros CAD; la mayor parte de cuestiones relacionadas con el diseño están ligadas a la interfaz de usuario de la aplicación *TideCAD*:

4.1. Interfaz Gráfica de Usuario de *TideCAD*

La GUI de *TideCAD* está dividida en tres secciones principales: la primera, es la barra de menú. Debajo de ella, aparece la barra de herramientas compuesta por todas las acciones de edición que el usuario puede llevar a cabo en la aplicación; y por último, aparece el espacio de diseño, donde el usuario podrá visualizar y editar el contenido de los ficheros de dibujo.

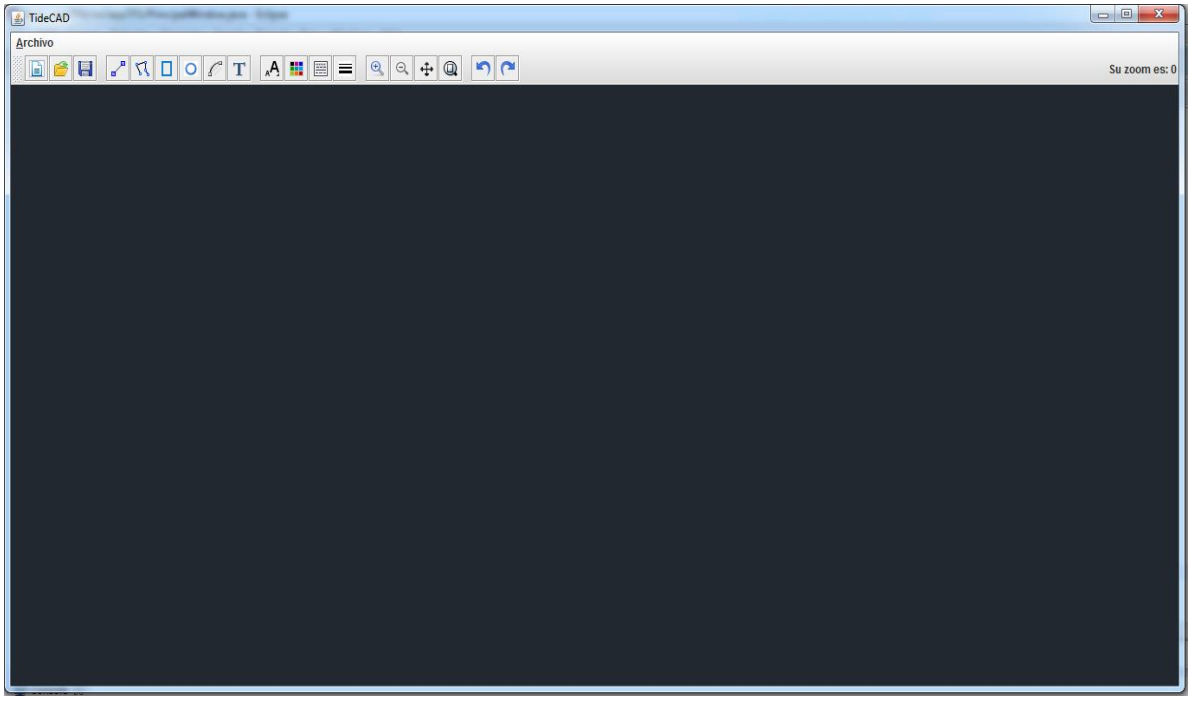

Ilustración 9: GUI de TideCAD

4.1.1. Barra de menú

La barra de menú contendrá un único ítem: “Archivo”. Al desplegarse, el usuario podrá realizar todas las acciones relacionadas con el tratamiento de los ficheros: Nuevo, Abrir, Guardar, Guardar Como y Exportar. Además, el menú ofrece la opción de cerrar la aplicación.

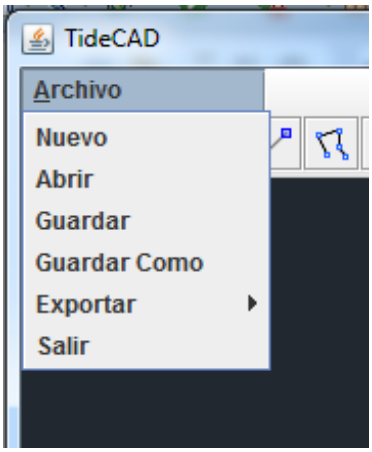

Ilustración 10: Menú Archivo

Para facilitar la interacción del usuario con la aplicación, se ha añadido una regla mnemotécnica para facilitar el despliegue del menú. Con la combinación de las teclas *Alt* y *“A”* se podrá extender el menú sin necesidad de hacer *click* sobre él.

A continuación, se procede a explicar cada uno de los elementos del menú:

4.1.1.1. Nuevo

Como se ha especificado en los RU y RS, el usuario no podrá crear un fichero de dibujo desde cero. Por tanto, la funcionalidad de este elemento es la de borrar todos las entidades, bien importadas por la apertura de un fichero de dibujo anterior o añadidas por el usuario, del espacio de diseño para que este quede vacío.

Si hay un fichero ya abierto y éste ha sido modificado, se mostrará el siguiente mensaje que permitirá al usuario guardar el fichero antes del borrado de todas sus entidades del espacio de diseño:

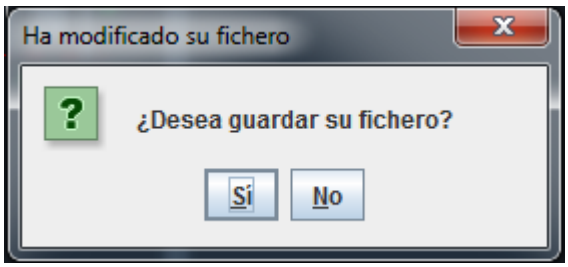

Ilustración 11: Ventana emergente para guardar el fichero

4.1.1.2. Abrir

Este elemento abrirá una ventana contextual en la que podremos navegar por los directorios de nuestro ordenador personal y seleccionar el fichero en formato DXF que queremos mostrar en la aplicación. Esta ventana sólo mostrará los ficheros que tengan este formato. Por otro lado, de los ficheros se mostrarán únicamente las entidades especificadas en los RU y RS.

Al igual que en el elemento anterior, si ya hay un fichero previo abierto y éste se ha modificado, la aplicación mostrará la ventana emergente de la *Ilustración 11*, para que el usuario pueda salvar sus cambios antes de abrir otro fichero.

4.1.1.3. Guardar

Permite guardar las modificaciones realizadas sobre el fichero de dibujo previamente abierto.

4.1.1.4. Guardar Como

Permite guardar las modificaciones realizadas sobre el fichero de dibujo previamente abierto en otro nuevo fichero manteniendo el fichero

original inalterado. Abrirá una nueva ventana en la que podremos seleccionar la ruta en la que queramos almacenar nuestro fichero y añadir el nombre deseado para este nuevo fichero sin necesidad de escribir la extensión, puesto que la aplicación lo guardará en formato DXF automáticamente.

4.1.1.5. Exportar

Este elemento contiene una serie de submenús, en los que podremos seleccionar el formato de acuerdo a lo especificado en los RU y RS en el que queremos exportar nuestro fichero de dibujo, modificado o sin modificar, previamente abierto.

Ilustración 12: Elemento "Exportar" desplegado

4.1.1.6. Salir

Permite cerrar la ventana de la aplicación, al igual que el botón rojo con la cruz blanca que trae el sistema operativo *Windows* por defecto para cerrar ventanas. Como ya se ha comentado en los elementos *Nuevo* y *Abrir*, si el fichero se ha modificado, antes de salir de la aplicación aparecerá la ventana emergente que podemos observar en la *Ilustración 11.*

4.1.2. Barra de herramientas

Para las funcionalidades de la aplicación relacionadas con la edición de los ficheros de dibujo se ha optado por la utilización de botones con iconos identificativos de cada una de ellas. Se ha optado por esta opción porque es más sencilla y más rápida la interacción del usuario a la hora de realizar modificaciones sobre los dibujos. Al final de esta barra, se muestra un mensaje indicando al usuario el nivel del zoom en el que se encuentra en ese momento.

Se han organizado los botones de la barra de herramientas en cinco categorías:

4.1.2.1. Tratamiento del fichero

Ilustración 13: Botones barra de herramientas 1

Los botones de esta sección se corresponden con varias de las operaciones que podemos realizar sobre los ficheros de dibujo. Más concretamente, las operaciones son las de *Nuevo, Abrir* y *Guardar*, que ya han sido explicadas en el apartado 4.1.1.

4.1.2.2. Añadir las entidades

Ilustración 14: Botones barra de herramientas 2

Los botones de esta sección de la barra de herramientas se corresponden con las posibles entidades que le podemos añadir a nuestros ficheros de dibujo. Las entidades de izquierda a derecha, tal y como se muestran en la *Ilustración 14*, son: *Línea, Polilínea,*

Rectángulo, Círculo, Arco y *Texto*. La forma en la que se añaden cada una de estas entidades se explicará en el Anexo C, Manual de Usuario.

4.1.2.3. Editar características de las entidades

Ilustración 15: Botones barra de herramientas 3

Los botones de esta sección se corresponden con cada una de las posibles modificaciones que se les puede realizar a las entidades. De izquierda a derecha según aparecen en la *Ilustración 15*¸ la funcionalidad de los botones es la siguiente:

- Fuente, tamaño y forma de las entidades de tipo "Texto": se abrirá una ventana emergente en la que podremos seleccionar el tipo de fuente, el tamaño y el estilo en la que la queremos.

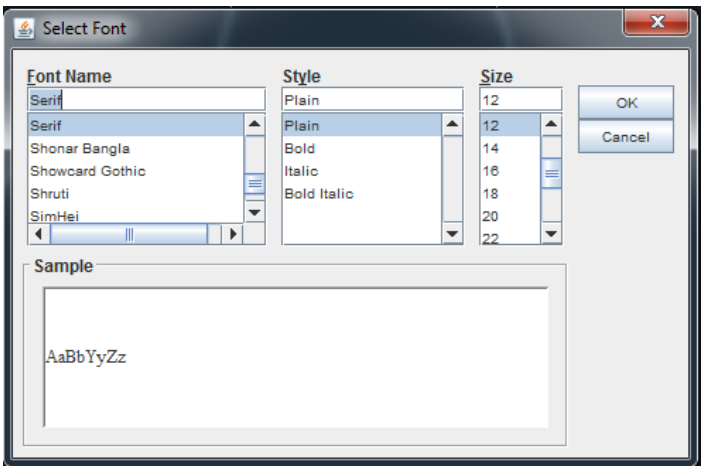

Ilustración 16: Ventana para elegir características de la fuente

- Color: se abrirá una ventana emergente en la que podremos seleccionar los colores especificados en los RU y RS de una lista desplegable. El motivo de la elección de estos colores y no otros es porque otras herramientas CAD, como *AutoCAD*, emplean esta lista de colores, identificando cada uno de ellos con un número entero. Por tanto, es necesario mantener esta

asociación número-color para asegurar la compatibilidad con estas otras aplicaciones.

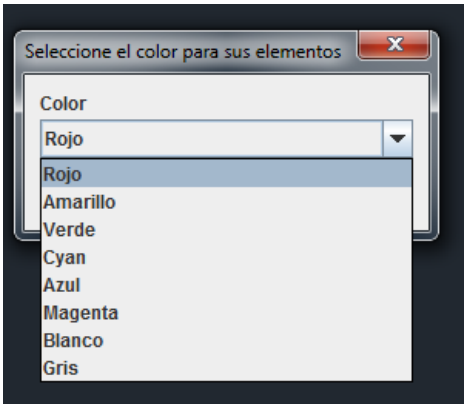

Ilustración 17: Ventana para elegir el color de las entidades

- Continuidad: al igual que con el color, aparecerá una ventana emergente en la que podremos elegir el valor *Continuo* o *Discontinuo* según queramos que aparezcan las líneas que conforman las entidades a añadir.

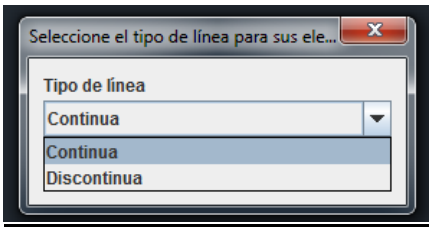

Ilustración 18: Ventana para elegir la continuidad de las entidades

- Grosor: aparece una ventana emergente en la que podemos elegir entre cuatro posibles valores, tal y como se especificó en los RU y RS, el grosor de las líneas que conforman las entidades a añadir. La decisión de poner cuatro niveles de grosor se debe a que se ha pensado que con esta cantidad de niveles es suficiente para satisfacer todas las necesidades de diseño en un caso real.

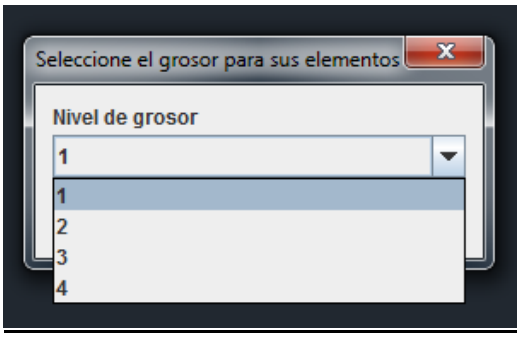

Ilustración 19: Ventana para elegir el grosor de las entidades

4.1.2.4. Representación del dibujo

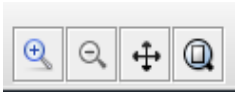

Ilustración 20: Botones barra de herramientas 4

Los botones de esta sección se corresponden con cada una de las transformaciones que podemos hacer en el espacio de diseño. De izquierda a derecha según aparecen en la *Ilustración 20*¸ la funcionalidad de los botones es la siguiente:

- Ampliar y reducir *zoom*: el número máximo de niveles de zoom será dos, tanto para ampliar como para reducir, con respecto al nivel de zoom original. Se ha decidido así por cuestiones de resolución y porque se ha pensado que era suficiente para satisfacer todas las necesidades del usuario. En cada nivel, las dimensiones de cada entidad se aumentan al doble o reducen a la mitad, con respecto al nivel anterior. Cuando se llegue al límite, el sistema avisará de ello con los siguientes mensajes:

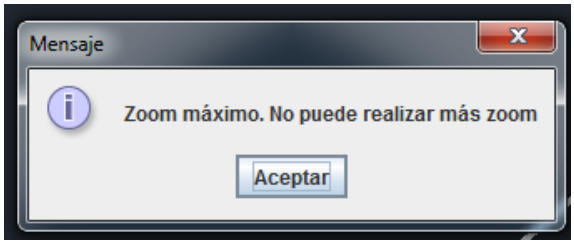

Ilustración 21: Mensaje de zoom máximo permitido.

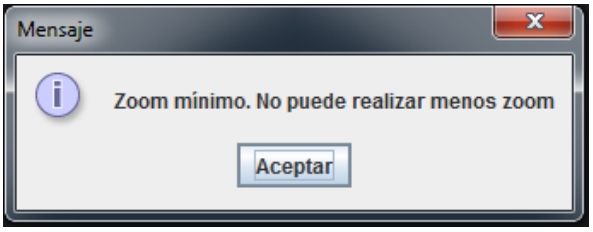

Ilustración 22: Mensaje de zoom mínimo permitido.

- Desplazarse por el dibujo: utilizando las flechas del teclado, el usuario puede desplazarse por el dibujo. Esta opción es muy útil tras realizar *zoom* de ampliación, en el que varias de las partes del dibujo se muestran fuera de los límites del espacio de diseño.

- Ajustar a la pantalla: tras pulsar en esta opción, el dibujo se mostrará con nivel de *zoom* 0 y situado de tal manera que todas las secciones del mismo son visibles. Si se ha realizado *zoom* o se ha hecho un desplazamiento por el dibujo, es necesario pulsar sobre esta opción antes de poder editarlo. De esta manera, se asegura que al insertar las nuevas entidades no haya problemas en cuanto a su dimensión y posición.

4.1.2.5. Deshacer y rehacer

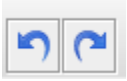

Ilustración 23: Botones barra de herramientas 5

Los botones de esta sección se corresponden con las funcionalidades que permiten al usuario el poder deshacer la última acción hecha, en este caso, eliminar la última entidad añadida; o volver a rehacer una acción desecha.

Esta funcionalidad solo se aplica a las operaciones que permiten añadir entidades y no a las operaciones de representación del fichero,

puesto que se ha considerado que en estas últimas se puede llegar a la situación anterior con relativa facilidad.

4.1.3. Espacio de diseño

En cuanto al espacio de diseño, al igual que otras aplicaciones CAD, se ha utilizado un color azul marino para el fondo para que se aprecien mejor los colores. Las entidades que añaden con color blanco, serán transformadas a color negro cuando se añadan a un fichero para evitar que coincidan con el color del fondo.

4.2. Alternativas de diseño

Como se ha especificado en la introducción de este apartado, ha sido Java el lenguaje elegido para la implementación de *TideCAD*. Sin embargo, existen lenguajes como C o C++, que también incluyen librerías gráficas, como por ejemplo *Qt*, sobre los que se podía haber implementado la aplicación.

La elección de este lenguaje frente a los otros se debe a que es el lenguaje con el que aprendí a programar y por tanto, con el que más cómodo me siento. Además, cuenta con el paradigma de POO que le aporta un toque de sencillez, seguridad y robustez. Por otro lado, como se ha comentado en el *Estado del Arte*, aunque el hecho de que pueda ejecutar sobre cualquier plataforma lo hace algo lento por la necesidad de interpretar el código, la idea de escribir un único código que pueda ejecutar sobre cualquier máquina es otro de los motivos por los que finalmente ha sido elegido. Por último, otro de los motivos con los que justificar la elección de Java frente a C es que no es necesario cerrar las referencias que no usemos gracias al recolector de basuras.

Una vez escogido Java, es necesario elegir una librería gráfica que permita la representación de los dibujos de los ficheros. Las dos principales opciones son *Java Swing* y S*WT*. La primera ya se ha comentado en el *Estado del Arte*, y la segunda además de lo comentado también en esa sección, es necesario decir que también

proporciona una funcionalidad similar a *Swing* para la implementación e interfaces gráficas. La elección, por tanto, de *Swing* frente a *SWT* se debe a que es más sencillo la utilización de una librería en el código que la necesidad de añadir un complemento al *framework* correspondiente. Además, el API de *SWT* es poco intuitivo.

Por último, entre las dos bibliotecas comentadas en el *Estado del Arte* para el tratamiento de los ficheros *dxf*, finalmente ha sido *Kabeja* la elegida debido a que cuenta con una API mejor, mayor distribución y mayor cantidad de ejemplos en la red.

5. IMPLEMENTACIÓN Y PRUEBAS

En este apartado se procede a comentar los detalles de la implementación de la aplicación *TideCAD* así como el conjunto de pruebas realizadas sobre el mismo para asegurar su correcto funcionamiento. La parte de implementación quedará dividida a su vez en dos partes: la primera para explicar el código de la interfaz del usuario y todo lo relacionado con ella y la segunda para hablar sobre la funcionalidad principal de la aplicación.

5.1. Implementación

El código se ha dividido en dos clases principales: *PrincipalWindow* y *LinesComponent*. La primera de ellas contiene todo el código relacionado con la interfaz y la funcionalidad de la aplicación, mientras que *LinesComponent,* por su parte, es una clase que extiende la clase *JComponent* de *Java Swing* y es sobre la que se dibujan todas las entidades de los ficheros y la que añaden los usuarios. Como se ha comentado en la introducción de este apartado, se va a separar el desarrollo de este apartado diferenciando la parte de la GUI y la parte de la funcionalidad de la aplicación, aunque se desarrollen en la misma clase:

5.1.1. Interfaz Gráfica

Para la implementación de la GUI se ha utilizado la librería gráfica *Java Swing*. En primer lugar, es necesario crear un hilo sobre el que ejecutar la aplicación. Este hilo se encargará de crear un objeto de esta clase, en cuyo constructor se llama al método que contiene toda la funcionalidad y que es el que realmente crea la aplicación.

Como se ha explicado en el diseño, la ventana tiene tres partes diferenciadas: la barra de menú en la parte superior, la barra de herramientas con los botones que proporcionan la funcionalidad a la aplicación y el espacio de diseño donde se muestran las entidades. En este apartado se pretende explicar cuáles han sido los elementos de *Java Swing* empleados en cada parte.

En primer lugar, para el espacio de diseño se ha utilizado un *JComponent*, que se corresponde con un objeto de la clase *LinesComponent* y de la que se hablará más adelante.

Para la barra de menú de la parte superior se han utilizado los siguientes elementos:

- *JMenuBar*: es la barra de menú en sí. Se sitúa en la ventana y sobre ella se colocarán los siguientes elementos.

- *JMenu*: permite crear un menú vertical que se añade sobre la barra anterior. Se utilizado para crear el menú de "Archivo", y además, como se ha comentado anteriormente, se le ha añadido una regla mnemotécnica para facilitar la interacción con los usuarios.

- *JMenuItem*: conforma cada uno de los elementos del menú. Son todas las opciones del menú "Archivo" explicadas en el apartado de Diseño.

Por último, para la barra de herramientas utilizaremos *JToolBar* que nos permite añadir elementos en función del diseño establecido. Los elementos utilizados han sido botones de tipo *JButton*. A cada uno de estos botones se le ha añadido un icono para que el usuario pueda apreciar su funcionalidad utilizando el elemento *ImageIcon* de *Swing*. Para concretar la funcionalidad de estos botones, se utiliza un *ActionListener* sobre el que se desarrolla el código. Al final de esta barra se utiliza un *JLabel* para indicar al usuario el nivel de zoom establecido en ese momento.

5.1.1.1. *LinesComponent*

Como se ha explicado con anterioridad, esta clase se utiliza como un componente de la ventana donde se mostrarán las distintas entidades. *LinesComponent* contiene clases embebidas para cada una de las entidades que puede añadir el usuario con atributos identificativos de esa entidad y que posteriormente nos servirán para dibujarlos.

Además de estas clases, *LinesComponent* contiene una lista enlazada para almacenar cada una de las entidades que puede leer y puede añadir a los ficheros, así como métodos para interactuar con estas listas. También tiene variables globales para controlar aspectos como el *zoom*, el *pan* o las coordenadas del fichero abierto.

Pero sin duda lo más importante de esta clase es el método *Paint*. Este método recibe un único argumento que es un objeto de la clase *Java Graphics*. Dicho objeto nos proporciona métodos para dibujar las distintas entidades de los ficheros de dibujo. Además, nos permite aplicar transformaciones para los efectos de ampliación o desplazamiento del dibujo o el grosor y tipo de las líneas que conforman las entidades.

5.1.2. Funcionalidades

Para seguir un orden a la hora de exponer cómo se han implementado las distintas funcionalidades de la aplicación, se va a explicar de izquierda a derecha según aparezcan en la barra de herramientas. Las funcionalidades que aparezcan en el menú “Archivo”, pero no en la barra, se explicarán junto a las del primer grupo.

Ilustración 24: Barra de herramientas de TideCAD

5.1.2.1. Tratamiento del fichero

Nuevo: esta funcionalidad se encarga de borrar todas las entidades del espacio de diseño. Esto se consigue eliminando todo el contenido de las listas enlazadas previamente mencionadas. Esta funcionalidad, también borra los elementos de las listas donde se almacenan las entidades “hechas” y “deshechas” que se explicarán en este apartado.

Abrir: tras consultar una variable global se decide si el fichero ha sido modificado o no. En caso afirmativo se llamará a la funcionalidad de guardar que se explica a continuación. Tras esto, se abre una ventana para que el usuario pueda seleccionar los ficheros *dxf*. Se ha utilizado un objeto de la clase *JFileChooser*, incluida en el paquete de *Java Swing* para esta acción. Utilizaremos *FileInputStream* para obtener los bytes del fichero.

Una vez seleccionado el fichero, se obtendrán sus límites para la posterior transformación a las coordenadas del espacio de diseño y se irán recorriendo cada una de las capas, mediante el uso de iteradores [36], añadiendo sus entidades a las listas enlazadas de la clase *LinesComponent.* Cada una de estas listas enlazadas se recorrerá en el método *paint* y se representarán utilizando los métodos proporcionados por *Graphics*. Además, se mantiene una variable global almacenando la última capa del fichero para, posteriormente, añadir en ella las entidades.

Guardar: para guardar el fichero se utiliza *FileOutputStream* para convertir los bytes de nuevo a un fichero especificando como ruta la obtenida al abrir el fichero. Tras esto, se utiliza un objeto *Generator* proporcionado por *Kabeja* para generar el archivo en formato *dxf*.

Guardar Como: realiza el mismo procedimiento que la anterior salvo que, en vez de utilizar la ruta del fichero almacenada en su apertura, abre una ventana para que el usuario pueda elegir una nueva donde almacenar su fichero en formato *dxf*. Para esta ventana, al igual que en *Abrir*, se ha utilizado un objeto *JFileChooser*.

Exportar: para todos los formatos la implementación es similar: cada uno de los formatos de salida cuenta con un *Generator* específico al que se le pasa el documento y un "serializador" SAX, que permite convertir el objeto a bytes para luego recuperarlo y al que se le añade el *FileOutputStream* [37].

Salir: tras comprobar si se ha modificado el fichero, se utiliza la llamada al sistema *System.exit(0)* para terminar la ejecución del programa.

5.1.2.2. Añadir las entidades

Para añadir todas las entidades se utilizará una implementación similar: se utilizará un objeto *MouseListener* para capturar los eventos del ratón [38]. Este objeto forma parte de la librería *Java AWT*. Utilizamos estos eventos para capturar las coordenadas del ratón y calcular los diferentes parámetros de las entidades necesarios para dibujarlos posteriormente. Además, en función de cómo el usuario dibuje la entidad, explicado posteriormente en el Anexo C, será necesario utilizar también un *MouseMotionListener* que captura eventos del ratón mientras este está en movimiento [39].

Estos *Listeners* se añaden al espacio de diseño cuando se pulsa sobre el botón correspondiente. Además de añadir el correspondiente, es necesario eliminar los anteriores que hayan sido añadidos previamente para evitar conflictos.

Por último, se comprobará si se ha abierto un fichero y se creará un objeto de la entidad específica proporcionado por *Kabeja* que añadiremos a la última capa del fichero.

5.1.2.3. Editar características de las entidades

Elegir fuente: para la elección de la fuente de las entidades de tipo Texto se utilizará un objeto *JFontChooser* [1] que ofrece una ventana en la que el usuario puede seleccionar tipo, forma y tamaño de la fuente y que devolverá un objeto de tipo *Font*.

Elegir color: para la elección del color se utiliza *JOptionPane*, de *Swing,* que nos permite mostrar una ventana de diálogo con una lista desplegable de opciones. Una vez seleccionada la opción se actualizará una variable global de tipo *Color* y otra de tipo *int* (para las entidades del fichero).

[1] http://www.java2s.com/Code/Java/Swing-Components/JFontChooser.htm

Elegir continuidad: se utiliza el mismo estilo que para el color. En este caso se utiliza una variable global *booleana* al tener dos posibles valores.

Elegir grosor: se utiliza el mismo estilo que para el color y la continuidad. En este caso se utiliza una variable de tipo *int*.

5.1.2.4. Representación del dibujo

Zoom: para la posibilidad de ampliar o reducir el dibujo se realiza una transformación afín (objeto *AffineTransform*) de escalado [40]. En cada *click* sobre los botones se amplía/reduce las dimensiones del espacio de diseño al doble/mitad. Para activar esta transformación de escalado, se mantiene una variable *booleana* que se comprueba cada vez que se llama al método *paint*.

Desplazamiento: se aplica otra transformación afín, pero en este caso de traslación. La cantidad que se va desplazando es una variable que se va aumentando a medida que el usuario pulsa sobre las teclas del teclado. Al igual que con el *zoom*, se mantiene una variable de tipo *boolean* que se comprueba cada vez que se llama al método *paint*.

Ajustar a la pantalla: como se almacena en variables cuales son los niveles actuales de *zoom* y de desplazamiento, esta opción consiste en hacer las transformaciones inversas tanto de traslación como de escalado. También se mantiene una variable *booleana* para saber si se tiene que efectuar las transformaciones.

5.1.2.5. Deshacer y rehacer

Para la funcionalidad de deshacer y rehacer es necesario crear una pila en la clase principal donde se irán almacenando variables de tipo *String* donde se guardan los nombres de las entidades que se van creando. Cuando se pulsa sobre “Deshacer”, se extrae de esta pila la última acción hecha y se añade a otra pila donde se almacenan las acciones desechas. Si se pulsa sobre “Rehacer”, se realiza el proceso contrario.

Por otro lado, es necesario crear en *LinesComponent* otra lista enlazada por cada una de las entidades para ir almacenando las que se van borrando.

Por tanto, cuando se pulsa sobre “Deshacer” se comprueba cual fue la última entidad añadida en la pila y se mueve dicha entidad de la lista en la que se encontraba a la nueva creada para almacenar los elementos borrados. Si se pulsa sobre “Rehacer”, se recupera esta entidad de la nueva lista y se añade a la lista original.

5.2. Pruebas funcionales

En este apartado se va a especificar el Plan de Pruebas realizado para asegurar el correcto funcionamiento del sistema y el cumplimiento de todas las necesidades propuestas en este documento.

Para cada una de las pruebas se va a utilizar la siguiente tabla:

Identificador: PF-XX	
Descripción	
Estado inicial	
Estado final	
Procedimiento realizado	
Resultado	

Tabla 61: Plantilla especificación Pruebas Funcionales.

Cada uno de los campos de la tabla anterior se describe a continuación:

- Identificador: un código identificativo de cada prueba. Está compuesto por un código alfanumérico que indica que nos encontramos ante una prueba funcional (PF), seguida de un número identificativo de cada una de ellos.

- Descripción: breve descripción de la prueba a realizar.

- Estado inicial: conjunto de precondiciones que tienen que darse para la realización de la prueba.

- Estado final: descripción de la salida o producto de la prueba.

- Procedimiento realizado: descripción más detallada del modo de ejecución de la prueba.

- Resultado: indica el resultado de la prueba realizada. Podrá ser, "Éxito" o "Fallo".

Las pruebas realizadas sobre el sistema son:

Identificador: PF-01	
Descripción	Se prueba las funcionalidades abrir y guardar relacionadas con el tratamiento de los ficheros y añadir Línea.
Estado inicial	No es necesario realizar ninguna acción previa.
Estado final	La salida es el mismo fichero modificado con la línea añadida.
Procedimiento realizado	Se abre un fichero y se comprueba que se muestran las entidades especificadas. Se añade una línea y se guarda. Después se vuelve a abrir el fichero para ver que la línea se ha guardado correctamente.
Resultado	Éxito

Tabla 62: PF-01: Abrir, Guardar y añadir Línea.

Identificador: PF-02	
Descripción	Se prueba la funcionalidad Guardar Como y añadir Círculo.
Estado inicial	Hay un fichero previamente abierto.
Estado final	La salida es un nuevo fichero con el mismo dibujo y con el nuevo círculo.
Procedimiento realizado	Se añade un rectángulo al fichero abierto y se guarda con otro nombre en otra ruta. Después se vuelve a abrir el fichero para ver que el círculo se ha añadido correctamente.
Resultado	Éxito

Tabla 63: PF-02: Guardar Como y añadir Círculo.

Identificador: PF-03	
Descripción	Se prueba la funcionalidad Exportar en SVG y añadir Rectángulo.
Estado inicial	Hay un fichero previamente abierto.
Estado final	La salida es un nuevo fichero en formato SVG con el mismo dibujo y con la nueva entidad.
Procedimiento realizado	Se añade un rectángulo al fichero abierto y se exporta a otro formato. Después se comprueba con otra aplicación que la entidad se ha añadido correctamente.
Resultado	Éxito

Tabla 64: PF-03: Exportar SVG y añadir Rectángulo.

Identificador: PF-04	
Descripción	Se prueba la funcionalidad Exportar en JPG y añadir Polilínea.
Estado inicial	Hay un fichero previamente abierto.
Estado final	La salida es un nuevo fichero en formato JPG con el mismo dibujo y con la nueva entidad.
Procedimiento realizado	Se añade la polilínea al fichero abierto y se exporta a otro formato. Después se comprueba con otra aplicación que la entidad se ha añadido correctamente.
Resultado	Éxito

Tabla 65: PF-04: Exportar JPG y añadir Polilínea.

Identificador: PF-05	
Descripción	Se prueba la funcionalidad Exportar en PNG y añadir Arco.
Estado inicial	Hay un fichero previamente abierto.
Estado final	La salida es un nuevo fichero en formato PNG con el mismo dibujo y con la nueva entidad.
Procedimiento realizado	Se añade el arco al fichero abierto y se comprueba que no se pueden añadir arcos de más de 180º. Después se exporta a otro formato y se comprueba con otra aplicación que la entidad se ha añadido correctamente.
Resultado	Éxito

Tabla 66: PF-05: Exportar PNG y añadir Arco.

Identificador: PF-06	
Descripción	Se prueba la funcionalidad Exportar en PDF y añadir Texto.
Estado inicial	Hay un fichero previamente abierto.
Estado final	La salida es un nuevo fichero en formato PDF con el mismo dibujo y con la nueva entidad.
Procedimiento realizado	Se añade el texto al fichero abierto y se exporta a otro formato. Después se comprueba con otra aplicación que la entidad se ha añadido correctamente.
Resultado	Éxito

Tabla 67: PF-06: Exportar PDF y añadir Texto.

Identificador: PF-07	
Descripción	Se prueba la funcionalidad Exportar en TIFF y la de modificar el color de las entidades.
Estado inicial	Hay un fichero previamente abierto.
Estado final	La salida es un nuevo fichero en formato TIFF con el mismo dibujo y con la nueva entidad con el color modificado.
Procedimiento realizado	Se modifica el color de las entidades y se añade una línea al fichero abierto. Después se exporta a otro formato y se comprueba con otra aplicación que la entidad se ha añadido correctamente con el color adecuado.
Resultado	Éxito

Tabla 68: PF-07: Exportar TIFF y modificar colores.

Identificador: PF-08	
Descripción	Se prueba la funcionalidad de modificar el estilo, tamaño y forma de las entidades de tipo Texto.
Estado inicial	Hay un fichero previamente abierto.
Estado final	El dibujo se modifica añadiendo varias entidades de tipo Texto.
Procedimiento realizado	Se añaden varias entidades de tipo Texto en las que se varía su estilo, tamaño y forma (cursiva, negrita). Después se comprueba que se han añadido según las propiedades seleccionadas.
Resultado	Éxito

Tabla 69: PF-08: Modificar estilo, tamaño y forma de los textos.

Identificador: PF-09	
Descripción	Se prueba la funcionalidad de modificar la continuidad y el grosor de las líneas que conforman las entidades.
Estado inicial	Hay un fichero previamente abierto.
Estado final	El dibujo se modifica añadiendo varias entidades de diversos tipos.
Procedimiento realizado	Se añaden cuatro rectángulos utilizando los distintos valores para el grosor y dos círculos para probar los dos posibles valores para la continuidad.
Resultado	Éxito

Tabla 70: PF-09: Modificar continuidad y grosor de las entidades.

Identificador: PF-10	
Descripción	Se prueba la funcionalidad de ampliar el dibujo.
Estado inicial	Hay un fichero previamente abierto.
Estado final	Los detalles del dibujo quedan ampliados.
Procedimiento realizado	Se amplía el espacio de diseño dos niveles y se comprueba que no se permite más. También se comprueba que se modifica la etiqueta del nivel de zoom.
Resultado	Éxito

Tabla 71: PF-10: Funcionalidad de ampliar zoom.

Identificador: PF-11	
Descripción	Se prueba la funcionalidad de reducir el dibujo zoom.
Estado inicial	Hay un fichero previamente abierto.
Estado final	Los detalles del dibujo quedan reducidos.
Procedimiento realizado	Se reduce el espacio de diseño dos niveles y se comprueba que no se permite más. También se comprueba que se modifica la etiqueta del nivel de zoom.
Resultado	Éxito

Tabla 72: PF-11: Funcionalidad de reducir zoom.

Identificador: PF-12	
Descripción	Se prueba la funcionalidad de desplazarse por el dibujo.
Estado inicial	Hay un fichero previamente abierto.
Estado final	El dibujo queda desplazado.
Procedimiento realizado	Se desplaza el dibujo en las cuatro direcciones utilizando las flechas del teclado.
Resultado	Éxito

Tabla 73: PF-12: Funcionalidad de desplazarse por el dibujo.

Identificador: PF-13	
Descripción	Se prueba la funcionalidad de ajustar el dibujo a la pantalla.
Estado inicial	Hay un fichero previamente abierto sobre el que se han aplicado las funcionalidades del zoom y desplazamiento.
Estado final	El dibujo queda centrado en la pantalla como si no se hubiera aplicado transformación alguna.
Procedimiento realizado	Se pulsa sobre el botón de Ajustar pantalla y se comprueba que se restaura a la posición inicial.
Resultado	Éxito

Tabla 74: PF-13: Funcionalidad de ajustar el dibujo a la pantalla.

Identificador: PF-14	
Descripción	Se prueba la funcionalidad Deshacer.
Estado inicial	Hay un fichero previamente abierto.
Estado final	El dibujo queda tal y como se abrió tras eliminarse las entidades añadidas.
Procedimiento realizado	Se añaden una serie de entidades sobre el dibujo y se pulsa tantas veces sobre el botón “Deshacer” como entidades se añadieron. Todas las entidades se borran.
Resultado	Éxito

Tabla 75: PF-14: Funcionalidad de Deshacer.

Identificador: PF-15	
Descripción	Se prueba la funcionalidad Rehacer.
Estado inicial	Hay un fichero previamente abierto sobre el que se han añadido un número de entidades y se han eliminado todas mediante la funcionalidad “Deshacer”.
Estado final	El dibujo queda con las entidades añadidas de nuevo.
Procedimiento realizado	Se pulsa tantas veces sobre el botón “Rehacer” como entidades se eliminaron. Todas las entidades aparecen de nuevo en el dibujo.
Resultado	Éxito

Tabla 76: PF-15: Funcionalidad de Rehacer.

5.2.1. Matriz de trazabilidad RU → PF

Para asegurarnos de que ningún RU, excepto los inversos, queda sin comprobarse, se realiza una matriz de trazabilidad para asegurar que cada RU se valida en al menos una prueba funcional:

	PF-01	PF-02	PF-03	PF-04	PF-05	PF-06	PF-07	PF-08	PF-09	PF-10	PF-11	PF-12	PF-13	PF-14	PF-15
RUC-01	X														
RUC-02	X	X	X	X	X	X									
RUC-03	X														
RUC-04		X													
RUC-05			X	X	X	X	X								
RUC-06							X								
RUC-07									X						
RUC-08									X						
RUC-09								X							
RUC-10										X	X				
RUC-11												X			
RUC-12													X		
RUC-13														X	
RUC-14															X
RUR-01	X														
RUR-02	X	X	X	X	X	X									
RUR-03	X														
RUR-04													X		
RUR-05			X	X	X	X	X								
RUR-06							X								
RUR-07									X						
RUR-08									X						
RUR-09										X	X				
RUR-10					X										
RUI-01															

6. GESTIÓN DEL PROYECTO

En este apartado se hace un seguimiento del proyecto, mostrando su planificación y el control de las distintas actividades y de los recursos humanos empleados a lo largo del mismo. Además, se estipulará un presupuesto aproximado del proyecto.

En primer lugar se hablará de la metodología empleada a lo largo del mismo, indicando cuáles han sido las fases por las que ha ido pasando el proyecto. Además, se realizará una secuenciación de estas tareas especificando cuáles han sido las más críticas.

6.1. Metodología

La metodología que se ha empleado como base para la realización de este proyecto ha sido MÉTRICA v3. Esta metodología nos permite la planificación, desarrollo e implantación de sistemas de información. Promovida por el Ministerio de Hacienda y Administraciones Públicas, puede ser utilizada libremente con la única restricción de citar la fuente de su propiedad intelectual [41].

Este proyecto se va a centrar sobretodo en la fase de planificación y desarrollo, puesto que al ser la aplicación un prototipo no es necesario proporcionar tareas de mantenimiento. La Planificación del Sistema de Información (PSI) servirá para detallar el alcance del sistema y para elaborar un calendario de ejecución.

Las principales tareas de la fase de desarrollo de MÉTRICA v3 son: [41]

- Estudio de la Viabilidad del Sistema (EVS): consiste en analizar el conjunto de necesidades que tiene una organización para proponer una solución a corto plazo. Esta solución debe tener en cuenta las restricciones económicas, técnicas, legales y operativas.

- Análisis del Sistema de Información (ASI): se basa en obtener una especificación detallada de aquello que se va a construir. Además, servirá como base para la posterior fase de diseño.

- Diseño del Sistema de Información (DSI): en esta etapa se resuelve el problema descrito en la fase de análisis. Además, se define completamente la arquitectura y los componentes del sistema.

- Construcción del Sistema de Información (CSI): en este proceso se genera el código de los componentes del sistema de información y se desarrollan todos los procedimientos de operación para asegurar el correcto funcionamiento del sistema.

- Implantación y Aceptación del Sistema (IAS): su objetivo es la entrega y aceptación final del sistema en su totalidad.

6.2. Ciclo de vida

El ciclo de vida define el orden de las tareas o actividades que se realizarán. Para ordenar dichas tareas se ha utilizado un modelo de cascada retroalimentada, donde para que una etapa dé comienzo tiene que haber finalizado la anterior. Además, al ser retroalimentada, permite volver a etapas anteriores para revisarlas y modificarlas si fuese necesario.

Las distintas etapas que conformarán este ciclo de vida en el proyecto son:

- Análisis: en esta primera etapa se trata de capturar y describir todas las funcionalidades del sistema.
- Diseño: se plantea una solución para todo lo propuesto en la etapa de análisis relacionado con la aplicación.
- Codificación: se desarrolla el código de la aplicación. Surgen los primero prototipos sobre los que se pueden realizar las pruebas.
- Pruebas: se realiza una evaluación de la aplicación y se registran los resultados.

Las etapas de Planificación y Estudio de la Viabilidad se realizan una única vez al inicio del proyecto, por lo que no es necesario incluirlas en este ciclo de vida. Lo mismo ocurre con la etapa de Implantación, que se realiza una sola vez al final del mismo.

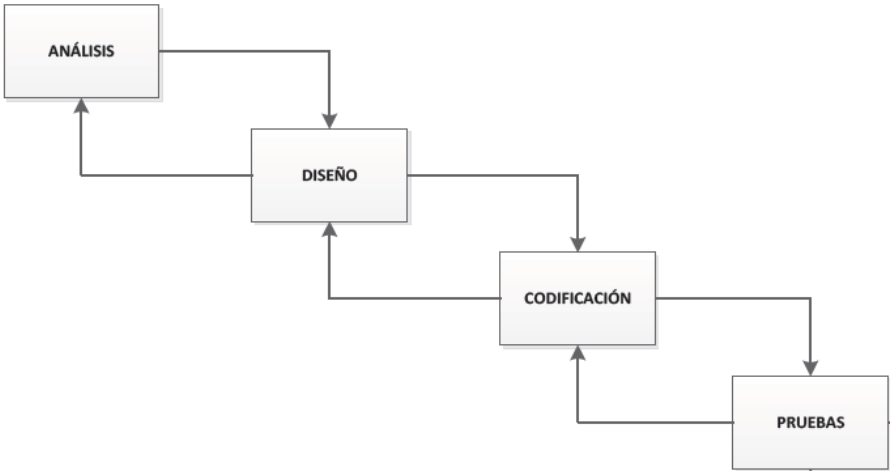

Ilustración 25: Ciclo de Vida del proyecto

6.3. Planificación temporal del proyecto

Según las etapas de la MÉTRICA v3 que finalmente se han optado por realizar en el proyecto según se ha indicado en el apartado 6.1, se procede a estimar cuál será la planificación a lo largo de los meses en los que se realizará el proyecto. Finalmente, se calcularán cuáles han sido las horas reales y se realizará una comparación con las estimadas. Para facilitar el entendimiento de la planificación, se utilizarán *Diagramas de Gantt* en los que podrá observar el desarrollo de las tareas con mayor claridad.

6.3.1. Planificación inicial estimada

El proyecto comenzará El 1 de Noviembre de 2015 y finalizará el 31 de Mayo de 2016. A lo largo de estos 213 días, se destinarán unas 360 horas para la realización del mismo repartidas de la siguiente manera:

- Planificación del Sistema de Información (PSI): 5 horas entre 1 de Noviembre y 8 de Noviembre.
- Estudio de la Viabilidad del Sistema (EVS): 15 horas entre 9 de Noviembre y 22 de Noviembre.

- Análisis del Sistema de Información (ASI): 70 horas entre 23 de Noviembre y 31 de Enero.
- Diseño del Sistema de Información (DSI): 60 horas entre 1 de Febrero y 20 de Febrero.
- Construcción del Sistema de Información (CSI): 170 horas entre 21 de Febrero y 30 de Abril.
- Implantación y Aceptación del Sistema (AIS): 30 horas entre 1 de Mayo y 31 de Mayo.

La justificación a por qué a partir de Febrero se realizarán aproximadamente la misma cantidad de horas en menos días es porque en el segundo cuatrimestre la carga académica es menor con respecto al primero.

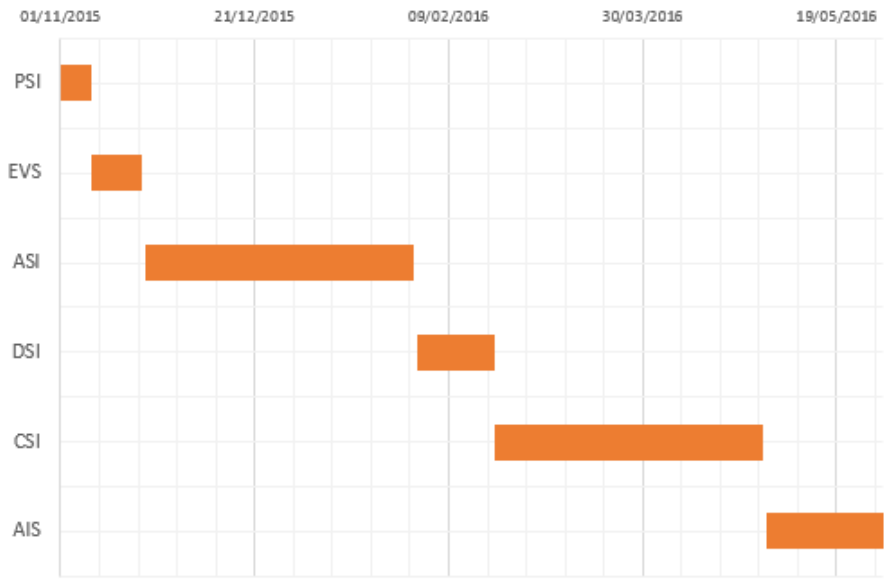

Ilustración 26: Diagrama de Gantt planificación inicial estimada

6.3.2. Planificación real final

Tras la realización del proyecto, se procede a comentar cuál ha sido la planificación real del mismo. Cómo se especificó en el Análisis del Sistema, por circunstancias del proyecto no se pudo hacer una toma de requisitos definitiva final, sino que a partir de una especificación inicial se han ido añadiendo una serie de requisitos nuevos en las distintas reuniones con el tutor. Por tanto, se podría decir que el desarrollo del proyecto ha sido un

proceso iterativo de las distintas etapas que conforman el ciclo de vida explicado en el epígrafe 6.2.

Cumpliendo los plazos iniciales y finales comentados en la planificación estimada, la realización y duración de las distintas tareas ha sido la siguiente:

- Planificación del Sistema de Información (PSI): 5 horas entre 1 de Noviembre y 8 de Noviembre.
- Estudio de la Viabilidad del Sistema (EVS): 12 horas entre 9 de Noviembre y 18 de Noviembre.
- Análisis del Sistema de Información (ASI) it.1: 25 horas entre 19 de Noviembre y 25 de Noviembre.
- Diseño del Sistema de Información (DSI) it.1: 15 horas entre 26 de Noviembre y 28 de Noviembre.
- Construcción del Sistema de Información (CSI) it.1: 17 horas entre 29 de Noviembre y 14 de Diciembre.
- Análisis del Sistema de Información (ASI) it.2: 17 horas entre 15 de Diciembre y 20 de Diciembre.
- Diseño del Sistema de Información (DSI) it.2: 23 horas entre 21 de Diciembre y 24 de Diciembre.
- Construcción del Sistema de Información (CSI) it.2: 25 horas entre 25 de Diciembre y 23 de Enero.
- Análisis del Sistema de Información (ASI) it.3: 15 horas entre 24 de Enero y 31 de Enero.
- Diseño del Sistema de Información (DSI) it.3: 16 horas entre 1 de Febrero y 10 de Febrero.
- Construcción del Sistema de Información (CSI) it.3: 75 horas entre 11 de Febrero y 17 de Marzo.
- Análisis del Sistema de Información (ASI) it.4: 12 horas entre 18 de Marzo y 21 de Marzo.
- Diseño del Sistema de Información (DSI) it.4: 21 horas entre 22 de Marzo y 28 de Marzo.
- Construcción del Sistema de Información (CSI) it.4: 87 horas entre 29 de Marzo y 15 de Mayo.
- Implantación y Aceptación del Sistema (AIS): 16 horas entre 16 de Mayo y 31 de Mayo.

Todas estas tareas hacen un total de 381 horas. Son 21 más que las que se estimaron inicialmente; pero se puede concluir que la planificación, en cuanto a horas, fue más o menos satisfactoria.

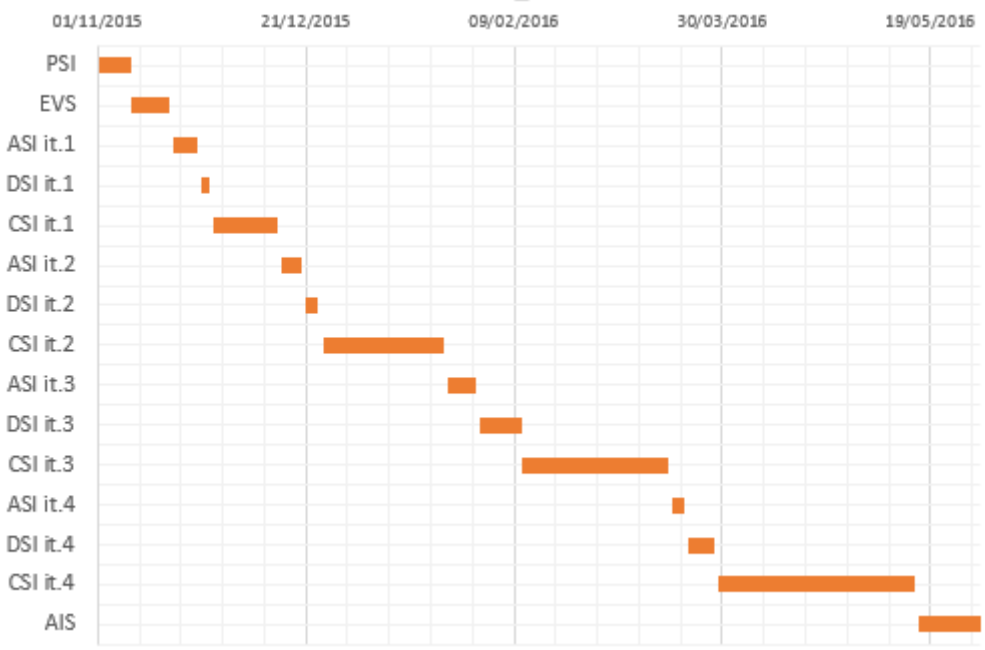

Ilustración 27: Diagrama de Gantt planificación final real

Las etapas más críticas han sido las de la iteración 4, puesto que se aproximaba el día de la entrega y aún quedaba trabajo por realizar.

6.4. Presupuesto

En este apartado se muestra el cálculo de costes para el desarrollo del proyecto y el presupuesto final del mismo. Además, se comenta el marco socio-económico sobre el que se desarrolla el proyecto.

6.4.1. Marco socio-económico

El objetivo del proyecto es el de diseñar e implementar un prototipo de aplicación de libre distribución con la que poder editar ficheros CAD. A día de hoy, *AutoCAD* es la aplicación más utilizada a la hora de realizar diseños en

ordenador. Sin embargo, *AutoCAD* es una aplicación de pago, por lo que muchas *startups* y empresas pequeñas de la Comunidad de Madrid y del resto de España, que no requieran de diseños extremadamente complejos, podrían utilizar *TideCAD* para solventar sus necesidades.

6.4.2. Costes de Software

Los costes relacionados con el *Software* quedan resumidos en la siguiente tabla:

Software	Unidades	Coste/unidad	Vida útil	Amortización	Coste Total	Coste aplicado al proyecto
Microsoft Windows 7 Professional	1ud	Gratis (Incluido en el coste del HW)	36 meses	-	0,00 €	0.00€
Eclipse Kepler	1ud	Gratis (Open Source)	36 meses	-	0,00€	0.00€
Kabeja 0.4	1ud	Gratis (Open Source)	36 meses	-	0,00€	0.00€
					Total	**0,00€**

Tabla 77: Costes de Software

6.4.3. Costes de Hardware

Los costes relacionados con el *Hardware* quedan resumidos en la siguiente tabla:

Hardware	Unidades	Coste/unidad	Vida útil	Amortización	Coste Total	Coste aplicado al proyecto
Acer Aspire v3-571g	1ud	549€	36 meses	15,25€/mes	549,00 €	106,75€
Toshiba Canvio Basics	1ud	51,95€	36 meses	1,44€/mes	51,95€	10,08€
Brother HL 5450DN	1ud	188,76 €	36 meses	5,24€/mes	188,76 €	36,68€
					Total	**153,51€**

Tabla 78: Costes de Hardware

6.4.4. Costes de materiales fungibles

Los costes relacionados con los materiales fungibles quedan resumidos en la siguiente tabla:

Hardware	Unidades	Coste/unidad	Coste Total	Coste aplicado al proyecto
Tóner para impresora Brother HL 5450DN	1ud	19,97€	19,97€	19,97€
Paquete de 100 folios DIN-A4	1ud	1,78€	1,78€	1,78€
Bolígrafos	5ud	0,25€	1,25€	1,25€
			Total	**23,00€**

Tabla 79: Costes de materiales fungibles

6.4.5. Costes de personal

Los costes relacionados con las personas responsables del proyecto quedan resumidos en la siguiente tabla. En este proyecto, al estar formado por un único integrante, es el encargado de desempeñar todos los roles.

Nombre	Rol	Coste/hora	Horas	Coste total
Diego García-Vaquero Fernández	Jefe de Proyecto	40,00€	50	2.000,00€
Diego García-Vaquero Fernández	Analista/Programador	25,00€	381	9.525,00€
			Total	**11.525,00€**

Tabla 80: Costes de personal

6.4.6. Presupuesto final

El presupuesto final queda definido de la siguiente manera:

Tipo de coste	Total
Costes de SW	0,00€
Costes de HW	153,51€
Costes de Materiales fungibles	23,00€
Costes de personal	11.525,00€
Gasto Total	**11.701,51€**

Margen de seguridad (10%)	1.170,15€
Beneficio (20%)	2.340,30€
PRESUPUESTO FINAL (sin IVA)	**15.211,96€**

Tabla 81: Presupuesto final

Teniendo en cuenta que en la fecha en la que se realizó el proyecto el IVA en España corresponde a un 21%, el presupuesto final del proyecto asciende a **18.406,71€** (DIECIOCHO MIL CUATROCIENTOS SEIS EUROS CON SETENTA Y UN CÉNTIMOS).

En dicho precio se incluyen los siguientes conceptos:

- Documentación relativa al proyecto.
- Código fuente.
- Derechos de propiedad intelectual y explotación.
- Derechos de distribución a terceros.

7. CONCLUSIONES Y TRABAJOS FUTUROS

En este apartado se comentarán las conclusiones obtenidas tras la realización del proyecto tanto en relación al producto obtenido, como personales. Además, se añadirá una sección en la que se indican los posibles aspectos que se pueden añadir a la aplicación de cara al futuro.

7.1. Conclusiones

Tras la elaboración del proyecto se puede decir que se ha conseguido el objetivo principal del mismo, que era la elaboración de un prototipo de una aplicación que nos permitiera la edición de ficheros de dibujo en formato *DXF*. *TideCAD* es una herramienta bastante completa que, pese a que no llega a ser una herramienta comercial, nos permite una funcionalidad básica y sencilla para los problemas más sencillos.

TideCAD no es un producto innovador, puesto que hay un sinfín de herramientas que ofrecen una funcionalidad muy superior, tanto de pago como gratuitas, y desde ese aspecto, es impensable la posibilidad de competir contra ellas. Sin embargo, es una aplicación realizada por un único programador, estudiante de su último año de carrera y usando tecnologías nunca antes vistas por él; aunque al fin y al cabo, eso es lo que se pretende con un Trabajo de Final de Grado: aprender.

Entrando más en el aspecto personal, este proyecto ha supuesto un reto para mí. Tal y como comenté en la introducción, el poder implementar una interfaz gráfica en Java era algo que rondaba por mi cabeza desde hacía un tiempo. Echando la vista atrás unos años y viendo cómo eran mis clases de Programación, en las que me costaba entender el concepto de bucle y de método; y viendo ahora lo que he conseguido con esta aplicación significa que ha habido un progreso desde la nada.

Sin embargo, el mundo de la programación es inmenso y este trabajo me ha servido, además de para aprender una gran cantidad de conceptos nuevos, para motivarme a seguir indagando en nuevas ideas con las que seguir formándome como ingeniero informático.

El nivel de dificultad al que me he enfrentado en la realización de este proyecto ha tenido dos niveles: uno para la realización de la GUI y otro para la funcionalidad de la aplicación. En lo relativo a la realización de la interfaz, creo que, pese a ser algo nuevo, me ha costado un poco menos debido a la cantidad de ejemplos y sitios web que hablan sobre ello y debido también a la utilización del *plug-in* de Eclipse mencionado con anterioridad, que permite suavizar la primera toma de contacto con *Java Swing*.

En cuanto a la funcionalidad de la aplicación, el nivel de dificultad ha sido considerablemente incrementado. *Kabeja* es una librería, hasta el momento, poco utilizada, por lo que el número de ejemplos sobre la tecnología que hay por la red es paupérrimo. Además, el código que proporciona de ejemplo en su sitio web no es correcto y da un sinfín de problemas. Con la clase *Graphics* de Java sí que he podido encontrar mayor documentación a la hora de realizar el trabajo.

En conclusión, creo que en este proyecto he podido demostrar todo lo aprendido durante estos cuatro años en cuanto a programación se refiere, y me ha servido para hacerme una idea aproximada del nivel con el que saldré al mercado laboral.

7.2. Trabajos futuros

TideCAD, al igual que la mayoría de aplicaciones de diseño, permite añadir un sinfín de nuevas funcionalidades con las que satisfacer las inquietudes que les vayan surgiendo a los usuarios. Sin embargo, considero que, las que explico a continuación, son las más relevantes y las que, por motivación personal, me plantee implementar:

- Añadir la posibilidad de importar ficheros de dibujo con otros formatos. Como se indicó en el *Estado del Arte*, *AuoCAD* ha liberado el formato DWG, por lo que sería una buena opción que la aplicación también pudiese importar estos ficheros.

- La posibilidad de seleccionar entidades dentro del dibujo y poder aplicarles una transformación específica, como rotación, modificación del grosor, etc.

- La posibilidad de convertir a *TideCAD*, además de en editor, en creador de ficheros de dibujo desde cero; porque como se ha especificado en los requisitos, la aplicación sólo permite editar los ficheros.

- La posibilidad de integrar más entidades a la aplicación, tanto para leer de ficheros como para ser añadidas.

- Añadir una barra de *scroll* en la parte derecha e inferior del espacio de diseño para facilitar el desplazamiento.

8. SUMMARY

In the following pages we can read a summary of this document. At first, we can see an introduction where I'm going to talk about the initial context of the project and the purpose of my teacher. In short, I'm going to talk about the motivation of the project. After that, there's a section where we can read the main and secondary objects of this project. Then, I'm going to explain what the results of the TFG are and finish this section with my personal conclusion and the future works I would like to do.

8.1. Introduction

After study the subject *Distributed Systems* with the tutor of this project, Félix García Carballeira, I turned to him to ask for an idea on which I can develop my Final Project. He told me about *Kabeja*, a Java Library that allows us to obtain information from CAD files in DXF format and to export those files to another formats. The idea was to implement a Graphic User Interface on which the user could open and handle different files in this format using Java as the programming language.

Java was the programming language with which, in the first course of the degree, we were taught to programme. It's true that the last years due to the speciality I've chosen, I have programmed more in programming languages like C or C++; but I feel more comfortable with Java. However, my level of Java is a basic level because we learnt it on the first years of the degree and it's a programming language that offers many possibilities and it's very complicated to domain it.

Java is an Object-oriented programming language which had its boom when its first commercial version was launched in 1996 by Sun Microsystems. Nowadays it's one of the most used programming languages around the world. [2] Java is a multiplatform language. It means that its packets contains the necessary resources that needs the operative system to run the java code. The way this is possible is because Java has an interpreter or virtual machine (JVM) for each of the platforms on which you want to run your code. However, in addition to the code, Java needs a set of libraries containing specific objects and classes of the platform. This

combination of virtual machine and libraries is known as Java Runtime Environment (JRE). [11]

In addition to the JRE Java has a garbage collector that avoids the problem of memory leaks which suffers other programming languages. Basically, the garbage collector is responsible for freeing the memory of the objects when there aren't references to them. [12]

One of the things that we couldn't see in that first years was the Graphic User Interfaces (GUI) for this programming language. We saw the theoretical concepts in the subject *User Interfaces*, but the exercises were about Web development. The topic of the GUIs in Java has been a topic that I have curiosity in and I have it pending to learn after the degree, so this was a big incentive to do this project.

In regard to the theme of the project, the possibility of opening and handling a DXF file, a format of CAD designs and used by *AutoCAD* and the rest of applications on the market, seemed a challenge which I would like to affront and to learn a lot of new and unknown issues until now. On the other hand, *AutoCAD* is the main tool used in engineering for drawing, so the idea of learning from it and implement a basic application which could obtain characteristics of *AutoCAD's* files was temptress.

8.2. Objectives

The main objective of the project is to develop a free software application able to open DXF files and show and modify its content with a range of entities that will be described in the following subparts. Furthermore, the application will can export the file, with modifications or not, to a number of standard formats like SVG, JPG, PNG, PDF and TIFF.

First, it proceeds to perform the design of the application, followed by the deployment of the code of the GUI. Then, it is developed all the functionality of the app's components to reach the main objective previously described.

In addition to the main objective, there is a number of secondary objectives associated with the project:

- Understanding the operation of a GUI in Java and learn how to implement it.
- To merge the code developed for the application with the functionality provided by Kabeja.
- Add to the application the basic functionality of a file editor: zoom, pan, undo, redo, among others.

8.3. Results

The result of this project is *TideCAD:* a free software prototype application with which the user could visualize drawing files and modify them. Then, it proceeds to explain the GUI of this application:

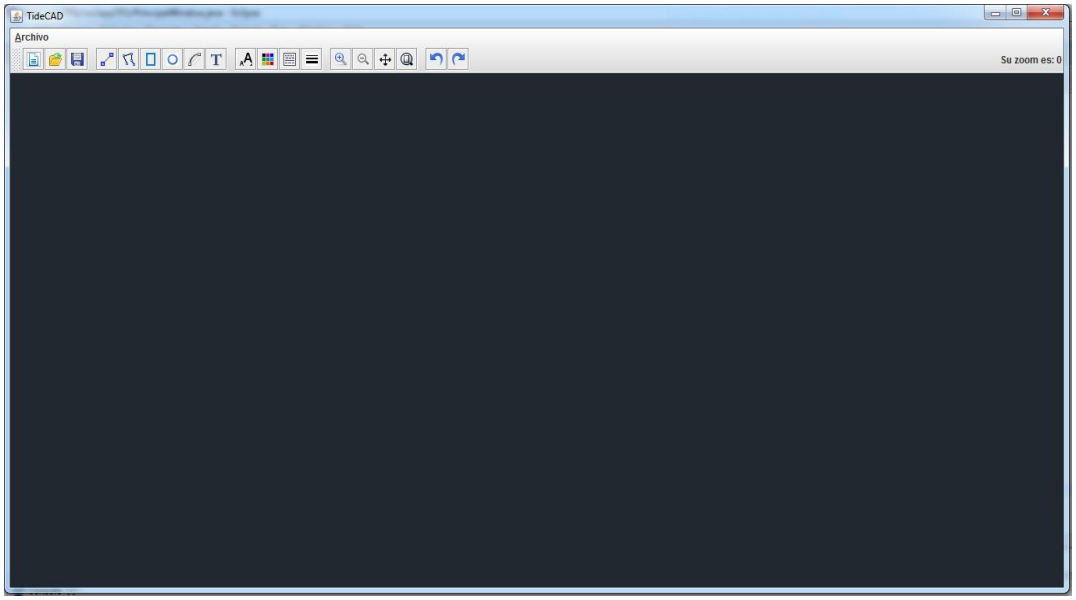

Figure 28: GUI of the application

First, we can see three distinct sections in the GUI. At the top, we can see a menu bar with only one item: *Archivo.* Under it, we can see another bar with buttons on which I've implemented the main functionality of the application. The last section, which is the bigger, is the drawing space where the user can visualize and edit the content of the drawing files.

In the item *Archivo* (File) of the menu bar we can found the following options:

- *Nuevo* (New): with this option we can clean our drawing space of all entities from the drawing files or added by us. This application is only an EDITOR. For this reason, we can't create a new blank drawing file. If we have modified a previously open file, the application allows us to save the file before cleaning the drawing space showing the following window:

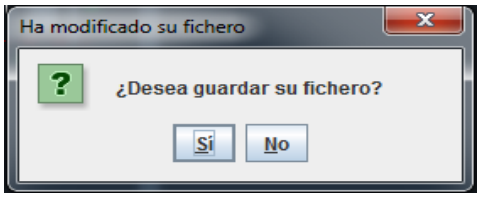

Figure 29: Pop-up window to save the file

- *Abrir* (Open): on this option we can import to the application an existing drawing file and visualize its content in the drawing space. At first, it is opened a pop-up window where the user can navigate between the folders of his personal computer and choose the file in DXF format which he wants to open. However, only the entities specified in the user requirements will be shown.

  As in the previous item, if there is already an open file before and this has been modified, the application will display the pop-up window of *Figure 29* and the user will can save his changes before opening another file.

- *Guardar* (Save): this option allows us to save the changes made to the drawing file previously opened.

- *Guardar Como* (Save As): this option allows us to save the changes made on the drawing file previously opened in another new file while maintaining the original file unchanged. It will open a new pop-up window in which we can select the path where we want to store our file. On this window we can add the desired name for this new file too without writing the extension, because the application will automatically save in DXF format.

- *Exportar* (Export): on this item we can choose between a number of sub items the format on which we want to export our drawing file with or without changes. The formats are specified in the user requirements and they are: SVG, JPG, PNG, PDF and TIFF.

- *Salir* (Exit): this option allows us to close the application window like the red button placed on the upper right corner in windows of *Windows OS*. Like *Open* and N*ew* items, if the file has been modified, before closing the window of the application, the pop-up window showed in *Figure 29* will be display.

Under this bar, how it is said before, there's another one which contains buttons that provide the functionality of the application. According to their function, they are grouped in five different sections along the bar.

The first group there are buttons related with the treatment of the files, so their functionality has been explained before.

In the next group of buttons, the user is able to add the entities specified in the user requirements. The name of this entities and how the user can add them is explained now:

- Line: to add this entity the user has to press the left button of the mouse in the position of the drawing space where he wants to start the line and release it where he wants to finish it.

- Polyline: for this entity, the user must go by clicking on the vertices he wants to add to it. Automatically, the application will join the last vertex added with the previous. Thus, it is shaping the entity. The user must press the Escape key on the keyboard to stop adding vertices.

- Rectangle: the user must press the mouse in the place where he wants the upper left corner of the rectangle and drag the pointer to the lower right corner. When the user reaches this point, he must release the mouse button.

- Circle: the way in which the circle is added is similar to the rectangle, in fact, an invisible square is drawn and inside of this square, the real circle

is inscribed. Therefore, the user must press the mouse in the upper left corner of this invisible square and drop it in the lower right.

- Arc: to draw the arc, the user must do three clicks: one to indicate where he wants to start the arc; the second to set the height and the third to indicate where it ends so the application can calculate the total angle.

- Text: to add this entity type, the user must click on the place where he wants to place the text. After this, a pop-up window in which the user will have to enter the text to be written and, after clicking the OK button, the text is added to the drawing space.

After this section, we have four buttons more to change the characteristics of the entities (color, thickness, continuity, etc):

- Font, size and style of Text entities: a pop-up window is displayed and the user can choose the name of the font, size and style for the Text entities.

- Color: a pop-up window is displayed and the user can select the colors specified in the user requirements from a drop-down list. The reason for choosing these colors and not others is because other CAD tools, such as AutoCAD, use this list of colors, identifying each with an integer. It is necessary to maintain this association number-color to ensure compatibility with these other applications.

- Continuity: as with color, a pop-up window in which the user can choose the Continuous or Discontinuous value is displayed.

- Thickness: as with color and continuity, a pop-up window in which the user can choose the value for the thickness is displayed. There are four possible values for this characteristic because this quantity of levels is enough for a real case.

The next section of buttons is related with the way of display the entities in the drawing space, that is, the transformations we can apply to this space:

- Zoom in and zoom out: the number of zoom levels is two, for expanding and for reducing, in relation to the original zoom level. The reason for this

quantity of levels is because it is enough to meet all user needs and by resolution issues. At each level, the dimensions of each entity are increased to double or halve, from the previous level. When the limit is reached, the system will warn about it with the following messages:

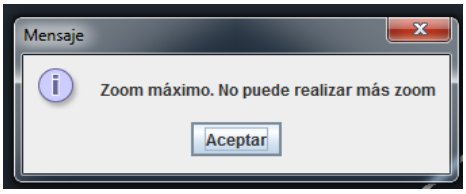

Figure 30: Message of maximum zoom allowed

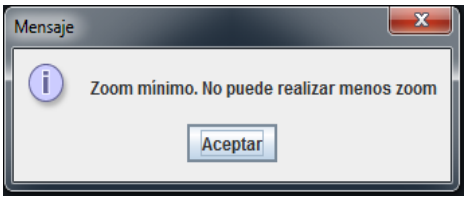

Figure 31: Message of minimum zoom allowed

- Scroll through the drawing: using the arrow keys, the user can scroll through the drawing. This option is useful after zooming, when several of the parts of the drawing are shown outside the boundaries of the drawing space.

- Fit to screen: the drawing is displayed with zoom level 0 and located so that all its sections are visible. If the user has done zoom or has made a move for the drawing, he needs to click on this option before he can edit it. So it ensures that when the user inserts the new entities there isn't problems in terms of size or position.

The last two buttons allow the user to undo the last action done, in this case, to remove the last added entity; or to redo an action discarded. On the far right of the bar, there is a tag which displays the zoom level on which the drawing is.

Finally, for the drawing space, as well as other CAD applications, is used a dark blue color for the background so that the colors can be more clearly.

The entities added with white color, will be transformed to black color when they are added to a file to avoid matching the background color of this file.

To conclude this section, it is good to say that *AutoCAD* is, nowadays, the most widely used application for making designs in a computer. However, AutoCAD is a paid application, so many startups and small businesses in the Community of Madrid and the rest of Spain, which don't require extremely complex designs, *TideCAD* could be used to meet their needs.

In the following table, we can see the breakdown of the different costs and the final budget:

Cost type	Total
Software costs	0,00€
Hardware costs	153,51€
Consumables costs	23,00€
Personal costs	11.525,00€
Total Cost	**11.701,51€**

Safety margin (10%)	1.170,15€
Benefit (20%)	2.340,30€
FINAL BUDGET (without VAT)	**15.211,96€**

Table 82: Final Budget

Considering that on the date on which the project was held, the VAT in Spain corresponds to 21%, the final budget for the project is € 18.406,71 (EIGHTEEN THOUSAND FOUR HUNDRED SIX EUROS AND SEVENTY ONE CENTS).

With that price, the following items are included:

- Documentation of the project.
- Source code.
- Intellectual property rights and exploitation.
- Distribution rights to third parties.

8.4. Conclusions and future works

This section describes the conclusions obtained after the completion of the project in relation to the product obtained and to my personal opinion too. In addition, there is a section where the possible aspects that can be added to the application in the future are discussed.

8.4.1. Conclusions

After the development of the project it can be said that has achieved the main objective of it, which was the development of a prototype of an application that would allow us to editing files in DXF drawing format. *TideCAD* is a very complete tool that, although it does not become a commercial tool, allows us a basic and simple functionality for easier problems.

TideCAD is not an innovative product, due to there are plenty of tools that offer superior functionality, both paid and free, and from that aspect, it is unthinkable to compete against them. However, it is an application made by a single programmer, student of the final year of his degree and using technologies never before seen by him; although after all, that is what is intended with a Final Project: to learn.

In terms of personal side, this project has been a challenge for me. As I mentioned in the introduction, to implement a GUI in Java it was something that haunted my head for some time. Looking back a few years and seeing how my programming classes were, when I had trouble understanding the concept of loop and method; and seeing now what I have achieved with this application means that there has been progress from nothingness.

However, the programming world is immense and this work has, in addition to learning a lot of new concepts, helped me to motivate me to continue work on new ideas to continue my education as a computer engineer.

The level of difficulty I've confronted in the realization of this project has two levels: one for the realization of the GUI and one for application functionality. With regard to the implementation of the interface, I think, despite being something new, it took me a little less effort because the number of examples and websites that talk about it and also because of the use of plug-in for Eclipse mentioned above, which can moderate the first contact with Java Swing.

As for the functionality of the application, the difficulty level has been considerably increased. *Kabeja* is a library, until now, very few used, so the number of examples of this technology there is very poor by the network. In addition, the sample code provided on its website is incorrect and provides a host of problems. With Java *Graphics* class I've had fewer problems because I could find more documentation when I was performing the work.

In conclusion, I believe that in this project I have been able to demonstrate what they have learned during these four years in terms of programming concerns, and has helped me to get a rough idea of the level that I will find in the job market.

8.4.2. Future works

TideCAD, like most of the design applications, can add an endless number of new features with which satisfy the concerns that worry users. However, I believe that the topics I am going to explain are the most relevant and perhaps by personal motivation I am going to implement:

- To add the ability of importing drawing files with other formats. As indicated in the State of the Art, *AutoCAD* has released the DWG format, so it would be a good option that the application could also import these type of files.

- The ability to select entities in the drawing space and to apply a specific transformation, such as rotation, modifying the thickness, etc.

- The possibility of converting *TideCAD* in, addition to editor, a file creator drawing from nothingness; because as specified in the user requirements, the application only allows us to edit drawing files.
- The possibility of integrating more features to the application and the quantity of formats to be added.

- To add a scroll bar on the right and bottom of the drawing space to facilitate scroll.

ANEXO A: ACRÓNIMOS Y GLOSARIO DE TÉRMINOS

Acrónimos y Siglas

2D: Dos dimensiones

3D: Tres dimensiones

API: *Application Programming Interface*

ASI: Análisis del Sistema de Información

AWT: *Abstract Window Toolkit*

BIM: *Building Information Modeling*

CAD: *Computer-Aided Design*

CSI: Construcción del Sistema de Información

CU: Caso de Usuario

DOM: *Document Object Model*

DSI: Diseño del Sistema de Información

DWT: *DraWing Template*

DXF: *Drawing eXchange File*

EVS: Estudio de la Viabilidad del Sistema

GUI: *Graphical User Interface*

HW: *Hardware*

IAS: Implantación y Aceptación del Sistema

IDE: *Integrated Development Environment*

IVA: Impuesto sobre el Valor Añadido

J2EE: *Java 2 Enterprise Edition*

J2ME: *Java 2 Micro Edition*

J2SE: *Java 2 Standard Edition*

JDBC: *Java Database Connectivity*

JDK: *Java Development Kit*

JDT: *Java Development Tools*

JPG: *Joint Photographic Group*

JRE: *Java Runtime Environment*

JVM: *Java Virtual Machine*

MIT: *Massachusetts Institute of Technology*

OTI: *Object Technology International*

PC: *Personal Computer*

PDF: *Portable Document Format*

PF: Prueba Funcional

PNG: *Portable Network Graphics*

POO: Programación Orientada a Objetos

PSI: Planificación del Sistema de Información

RSF: Requisito de Software Funcional

RSNF: Requisito de Software No Funcional

RU: Requisito de usuario

RUC: Requisito de Usuario de Capacidad

RUI: Requisito de Usuario Inverso

RUR: Requisito de Usuario de Restricción

SVG: *Scalable Vector Graphics*

SW: *Software*

SWT: *Standard Widget Toolkit*

TFG: Trabajo Fin de Grado

TIFF: *Tagged Image File Format*

XML: *eXtensible Markup Language*

Glosario de términos

Drag and drop: acción que consiste en mantener la tecla del ratón pulsada mientras se realiza un desplazamiento del mismo por la pantalla hasta que se suelta y genera como consecuencia una acción.

DWG: formato para los archivos de datos de *AutoCAD*.

Estándar de facto: estándar que no ha sido desarrollado por ninguna organización acreditada; pero que nacen a partir de productos de la industria con gran éxito en el mercado.

Fugas de memoria: error de SW que ocurre cuando un bloque de memoria reservada no es liberada por el programa.

Open-source community: comunidad que se encarga de tramitar el SW de código abierto, que consiste en publicar bajo una licencia de software el código fuente del producto.

Plug-in: complemento de una aplicación que permite agregarle una funcionalidad nueva.

Previsualización: proporciona una vision previa de un documento antes de que sea producido en su forma final.

Renderización: proceso mediante el que se genera una imagen o vídeo partiendo de un modelo en 3D.

ANEXO B: REFERENCIAS BIBLIOGRÁFICAS

[1] *R14 DXF Reference*. (1998). *Autodesk.com*. Recuperado en Abril de 2016, de http://www.autodesk.com/techpubs/autocad/acadr14/dxf/

[2] *¿Qué es Java? Concepto de programación orientada a objetos vs programación estructurada*. (2013). *Aprenderaprogramar.es*. Recuperado en Mayo de 2016, de http://www.aprenderaprogramar.es/index.php?option=com_content&view=article&id=368:ique-es-Java-concepto-de-programacion-orientada-a-objetos-vs-programacion-estructurada-cu00603b&catid=68:curso-aprender-programacion-Java-desde-cero&Itemid=188

[3] *Fundamentos de la POO*. (2008). *Msdn.microsoft.com*. Recuperado en Mayo de 2016, de https://msdn.microsoft.com/es-es/library/bb972232.aspx#XSLTsection127121120120

[4] Murguía, M. (2007). *¿Qué es un applet?* Universidad Nacional Autónoma de México. Recuperado en Mayo de 2016, de http://campus.iztacala.unam.mx/mmrg/mono54/archivos_archivos/arch_005_applets.pdf

[5] *Historia del lenguaje Java*. (2007). *Cad.com.mx*. Recuperado en Mayo de 2016, de http://www.cad.com.mx/historia_del_lenguaje_java.htm

[6] Gosling, J. (2007) *A brief history of the Green project*. Java.net. Recuperado en Mayo de 2016, de http://web.archive.org/web/20070127143602/http://today.java.net/jag/old/green/

[7] *Write once, run anywhere?* (2012). *ComputerWeekly*. Recuperado en Mayo de 2016, de http://www.computerweekly.com/feature/Write-once-run-anywhere

[8] *Oracle Technology Network for Java Developers | Oracle Technology Network | Oracle*. (2000). *Java.sun.com*. Recuperado en Mayo de 2016, de http://java.sun.com/j2se/

[9] *Oracle Technology Network for Java Developers | Oracle Technology Network | Oracle*. (2006). *Java.sun.com*. Recuperado en Mayo de 2016, de http://java.sun.com/javase

[10] *Java SE versions history*. (2012). *Codejava.net*. Recuperado en Mayo de 2016, de http://www.codejava.net/java-se/java-se-versions-history

[11] *Java Cafe: Que es Java y como funciona*. (2016). *Javaenejemplos.blogspot.com.es*. Recuperado en Mayo de 2016, de http://javaenejemplos.blogspot.com.es/2010/06/que-es-java-y-como-funciona-antes-del.html

[12] *Recolector de basura de java*. (2014). *Slideshare.net*. Recuperado en Mayo de 2016, de http://es.slideshare.net/softmasco/recolector-de-basura-de-java-garbage-collector

[13] M. Domínguez-Dorado, Guillermo Som. Todo Programación. Nº 11. Págs. 10-20. Editorial Iberprensa (Madrid). DL M-13679-2004. Agosto, 2005. *Imprimir desde Java y .NET*

[14] *JDK 5.0 Swing (Java Foundation Classes (JFC))-related APIs & Developer Guides -- from Sun Microsystems*. (2004). *Docs.oracle.com*. Recuperado en Mayo de 2016, de http://docs.oracle.com/javase/1.5.0/docs/guide/swing/

[15] Esteso, M. (2012). *Tutorial 14 Java: Swing (Interfaces gráficas)*. *Geeky Theory*. Recuperado en Mayo de 2016, de https://geekytheory.com/tutorial-14-java-swing-interfaces-graficas/

[16] *Graphics (Java Platform SE 7)*. (2008). *Docs.oracle.com*. Recuperado en Mayo de 2016, de https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html

[17] *Clase Graphics y sus métodos*. (2012). *Javaya.com.ar*. Recuperado en Mayo de 2016, de http://www.javaya.com.ar/detalleconcepto.php?codigo=130&inicio=40

[18] *What is Eclipse?* (2010). *SearchSOA*. Recuperado en Mayo de 2016, de http://searchsoa.techtarget.com/definition/Eclipse

[19] *A brief history of Eclipse*. (2007). *Ibm.com*. Recuperado en Mayo de 2016, de http://www.ibm.com/developerworks/rational/library/nov05/cernosek/

[20] Sanz, L. B., & Monreal, E. G. (2007). Kybele: Eclipse como IDE. *Universidad Rey Juan Carlos.*

[21] *Iniciándose en la plataforma Eclipse*. (2014). *Ibm.com*. Recuperado en Mayo de 2016, de https://www.ibm.com/developerworks/ssa/library/os-ecov/

[22] *SWT: The Standard Widget Toolkit*. (2003). *Eclipse.org*. Recuperado en Mayo de 2016, de https://www.eclipse.org/swt/

[23] *Eclipse Overview*. (2012). *Tutorialspoint*. Recuperado en Mayo de 2016, de http://www.tutorialspoint.com/eclipse/eclipse_overview.htm

[24] *WindowBuilder*. (2012). *projects.eclipse.org*. Recuperado en Mayo de 2016, de https://projects.eclipse.org/projects/tools.windowbuilder

[25] *Areas Técnicas*. (2002). *Unizar.es*. Recuperado en Mayo de 2016, de http://www.unizar.es/aeipro/finder/INGENIERIA%20DE%20PRODUCTOS/BF04..htm

[26] Kennedy, L. (2014). *A Brief History of AutoCAD | Scan2CAD*. *Scan2cad.com*. Recuperado en Mayo de 2016, de http://www.scan2cad.com/tips/autocad-brief-history/

[27] Hurley, S. (2008). AutoCAD Release History. *Between the Lines*. Recuperado en Mayo de 2016, de http://autodesk.blogs.com/between_the_lines/autocad-release-history.html

[28] García Sánchez, I. (2013). *Características y Ventajas principales de AUTOCAD*. *Sites.google.com*. Recuperado en Mayo de 2016, de https://sites.google.com/site/ivangarciasanchez90/objetivos/desarrollo-tema-7/1o

[29] *Ventajas y Desventajas de Autocad*. (2013). *Masterenautocad.com*. Recuperado en Mayo de 2016, de http://www.masterenautocad.com/cursoautocad/ventajas-y-desventaja-de-autocad/

[30] *About Kabeja*. (2006). *Kabeja.sourceforge.net*. Recuperado en Junio de 2016, de http://kabeja.sourceforge.net/

[31] Bah, T. (2009). *Inkscape: Guide to a Vector Drawing Program (Digital Short Cut)*. Pearson Education.

[32] *Kabeja News*. (2006). *Kabeja.sourceforge.net*. Recuperado en Junio de 2016, de http://kabeja.sourceforge.net/news.html

[33] *Ycad - Java CAD library.* (2014). *SourceForge*. Recuperado en Junio de 2016, de https://sourceforge.net/projects/ycad/

[34] Karlo, E. (2003). *Ycad 1.0.2 API*.

[35] *En el campo de batalla: C y Java*. (2007). *DeltaLinuXer*. Recuperado en Junio de 2016, de https://deltalinuxer.wordpress.com/2007/07/10/en-el-campo-de-batalla-c-y-java/

[36] *LayerSeparator.java in kabeja*. (2015). *Searchcode.com*. Recuperado en Diciembre de 2015, de https://searchcode.com/codesearch/view/15623735/

[37] *Using Library*. (2010). *Kabeja.sourceforge.net*. Recuperado en Diciembre de 2015, de http://kabeja.sourceforge.net/docs/devel/embedding.html

[38] *How to Write a Mouse Listener*. (2011). *Docs.oracle.com*. Recuperado en Febrero de 2016, de https://docs.oracle.com/javase/tutorial/uiswing/events/mouselistener.html

[39] *How to Write a Mouse-Motion Listener*. (2011). *Docs.oracle.com*. Recuperado en Febrero de 2016, de https://docs.oracle.com/javase/tutorial/uiswing/events/mousemotionlistener.html

[40] *AffineTransform (Java Platform SE 7)*. (2011). *Docs.oracle.com*. Recuperado en Abril de 2016, de https://docs.oracle.com/javase/7/docs/api/java/awt/geom/AffineTransform.html

[41] *PAe - Métrica v.3*. (2013). *Administracionelectronica.gob.es*. Recuperado en Junio de 2016, de

http://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html

ANEXO C: MANUAL DE USUARIO

El objetivo de este anexo es el de facilitar al usuario el primer contacto con la aplicación, así como guiarle paso a paso en su interacción con la misma.

La primera pantalla a la que se enfrenta el usuario nada más abrir la aplicación es esta:

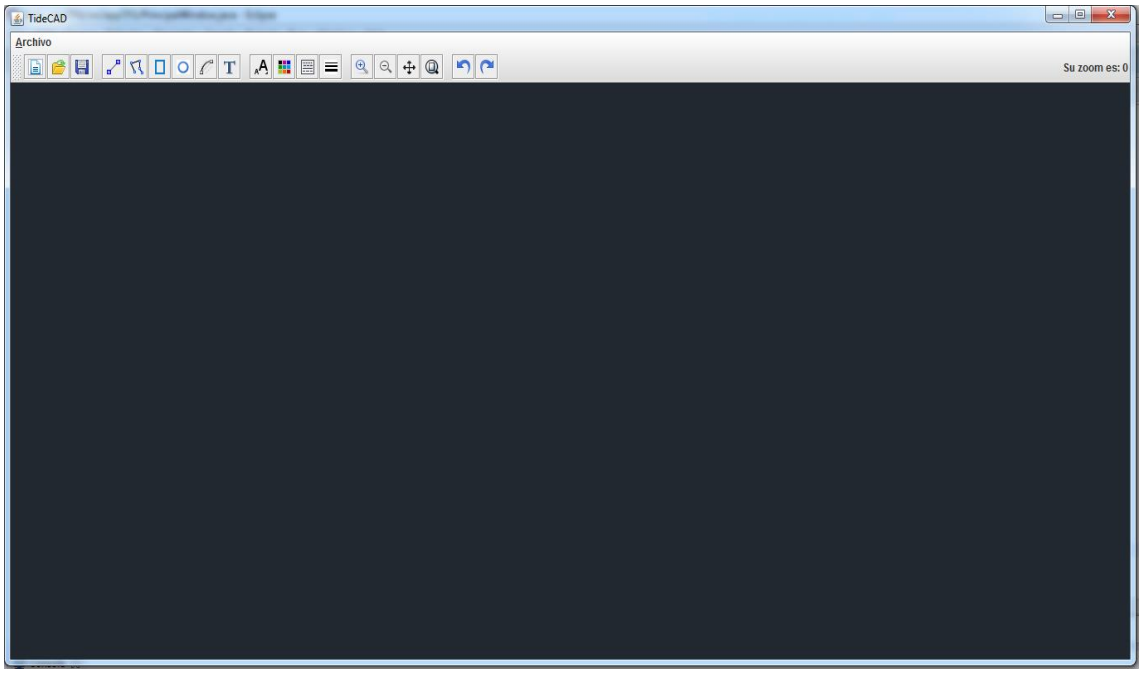

Ilustración 32: Pantalla principal TideCAD

Aquí el usuario tiene dos opciones: la primera, probar las funcionalidades de la aplicación, es decir, podrá añadir entidades, modificar sus características, ampliar, reducir o desplazarse por el espacio de diseño, etc. Sin embargo, como se ha especificado en los RU, **NO SE PODRÁ CREAR** un fichero de dibujo en *TideCAD.*

La segunda opción, a priori la opción elegida por los usuarios ya que se corresponde con la principal funcionalidad de la aplicación, es la de abrir un fichero de dibujo en formato DXF. El usuario tiene dos opciones para ello: a través de Archivo -> Abrir, o bien pulsando sobre el botón 

Tras esto, el usuario podrá añadir las diferentes entidades al fichero siguiendo los siguientes pasos para cada una:

- Línea: el usuario debe presionar el ratón donde quiere que comience la línea y soltar la tecla del ratón donde quiere que termine. Mientras mantiene pulsado, se irán dibujando las líneas intermedias hasta que finalmente suelte y se dibuje la línea definitiva.

- Polilínea: para esta entidad, el usuario deberá ir haciendo *click* sobre los distintos vértices que quiera añadir a la misma. Automáticamente, se irán uniendo el último vértice pulsado con el anterior para ir dando forma a la entidad. Para indicar que no se quiere seguir añadiendo vértices, se debe pulsar la tecla *Escape* del teclado.

- Rectángulo: el usuario debe presionar el ratón en el lugar donde quiere que se sitúe la esquina superior izquierda del rectángulo y arrastrar el puntero hacia la esquina inferior derecha. Cuando llegue a este punto, deberá soltar la tecla del ratón. Al igual que con la línea, se irán dibujando los rectángulos intermedios. Si se decide dibujar en otra dirección que no sea la indicada, el rectángulo no se dibujará.

- Círculo: la forma en la que se añade el círculo es similar a la del rectángulo, de hecho, se dibuja un cuadrado invisible sobre el que va inscrito el círculo dibujado. Por tanto, el usuario debe presionar el ratón en la esquina superior izquierda de este cuadrado y soltarlo en la inferior derecha. También se irán dibujando los círculos intermedios hasta que el usuario suelte, cuando se dibujará el definitivo.

- Arco: para dibujar el arco, el usuario debe realizar tres *clicks:* el primero para indicar donde quiere que comience el arco; el segundo para establecer la altura del mismo y el tercero y último para indicar dónde termina y así poder calcular el ángulo total del mismo.

- Texto: para añadir una entidad de tipo texto, se debe pulsar en el lugar donde se quiera situar el texto. Tras esto, aparecerá una ventana emergente en la que el usuario tendrá que introducir el texto a escribir y que, tras pulsar el botón *Aceptar* se añadirá al espacio de diseño.

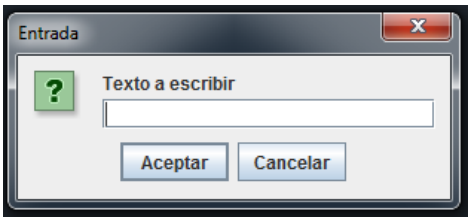

Ilustración 33: Ventana emergente para la introducción del Texto

Los siguientes botones sirven para modificar las características de las entidades. Todos crean nuevas ventanas con interfaces bastante intuitivas en las que el usuario podrá elegir los valores de los parámetros que quiere modificar. Dichas ventanas se muestran a continuación:

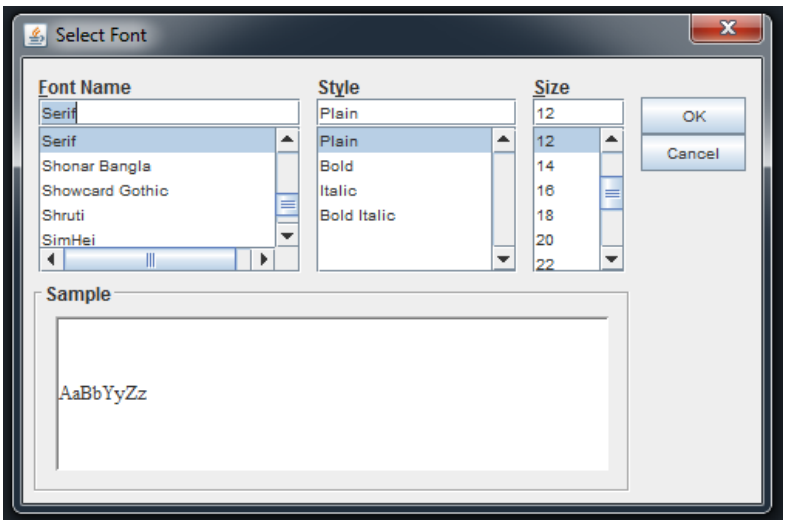

Ilustración 34: Selección de fuente, estilo y tamaño

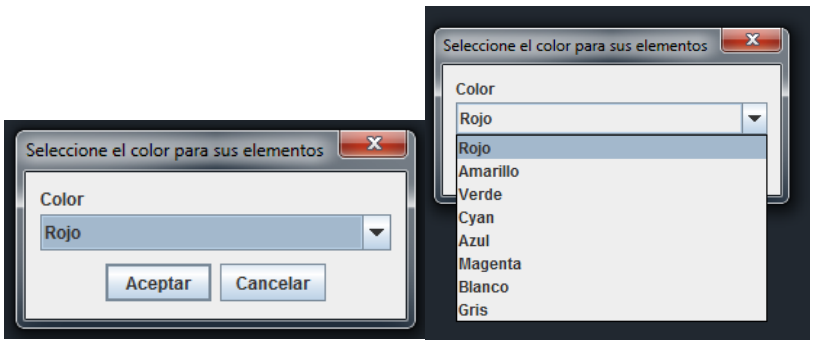

Ilustración 35: Selección de color de las entidades

Ilustración 36: Seleccionar continuidad de las entidades

Ilustración 37: Seleccionar grosor de las entidades

El usuario podrá también modificar el dibujo, ampliando o reduciendo los detalles del mismo, e incluso desplazarse por él.

Teniendo en cuenta que al abrir un fichero está a nivel de zoom 0, el usuario podrá ampliar hasta el nivel 2 y reducir hasta el -2. Para pasar de un nivel a otro de zoom será necesario pulsar sobre el botón correspondiente. El usuario podrá observar en todo momento el nivel de zoom en el que se encuentra.

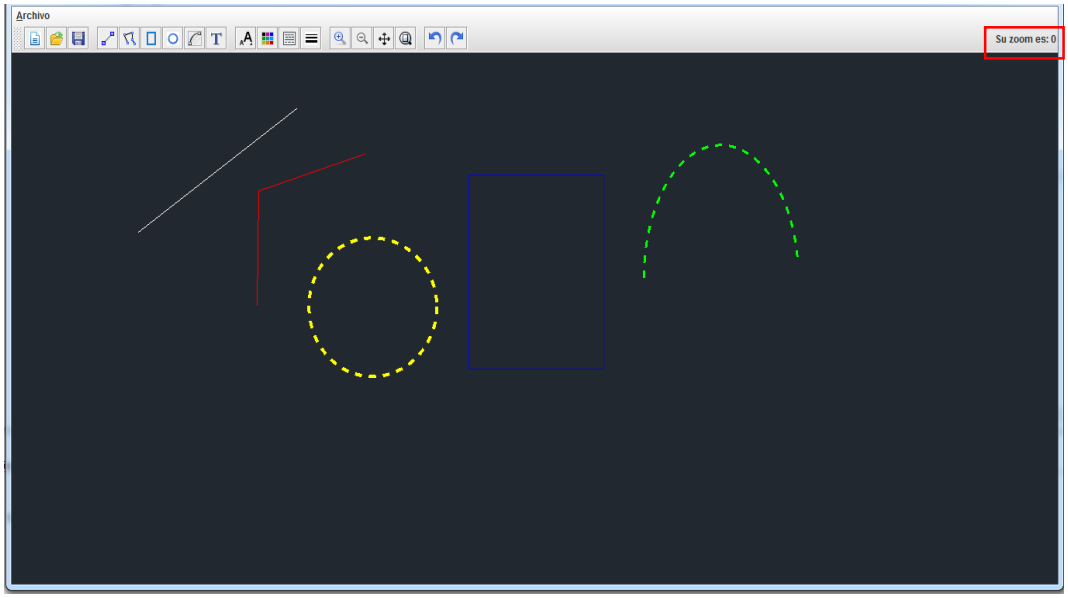

Ilustración 38: Ejemplo de dibujo con zoom 0

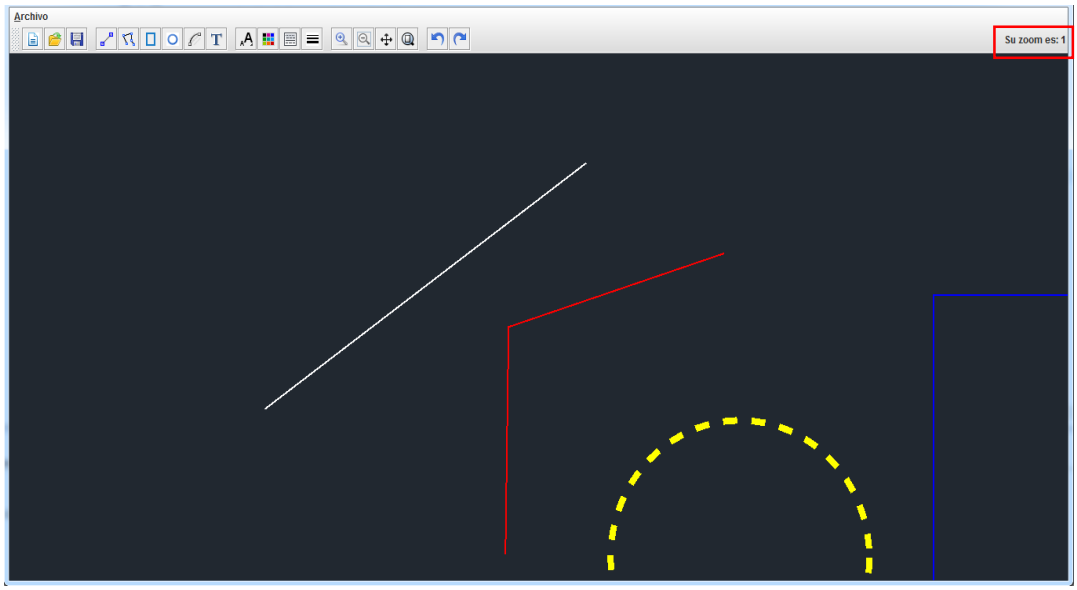

Ilustración 39: Ejemplo de dibujo con zoom 1

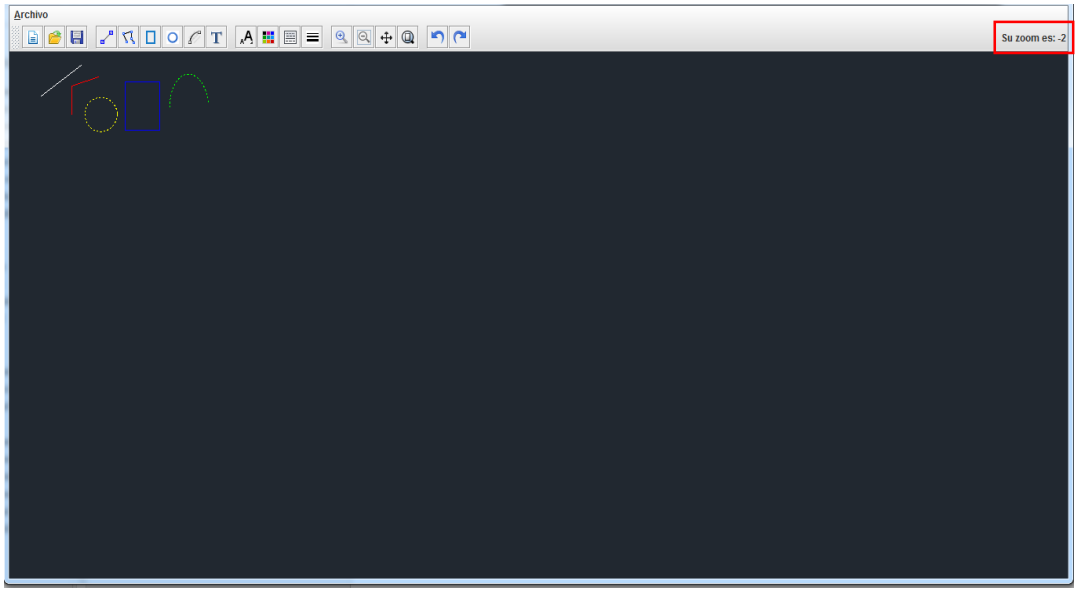

Ilustración 40: Ejemplo de dibujo con zoom -2

El usuario puede desplazarse por el dibujo utilizando las flechas del teclado. Lo que en realidad mueve el usuario es la vista desde la que se ve el dibujo, es decir, si pulsa la tecla de la derecha es porque querrá ver en el espacio de diseño las entidades que se encuentran fuera de sus límites por la parte derecha, por lo que el resto de entidades se moverán hacia la izquierda. Así con el resto de posibles desplazamientos.

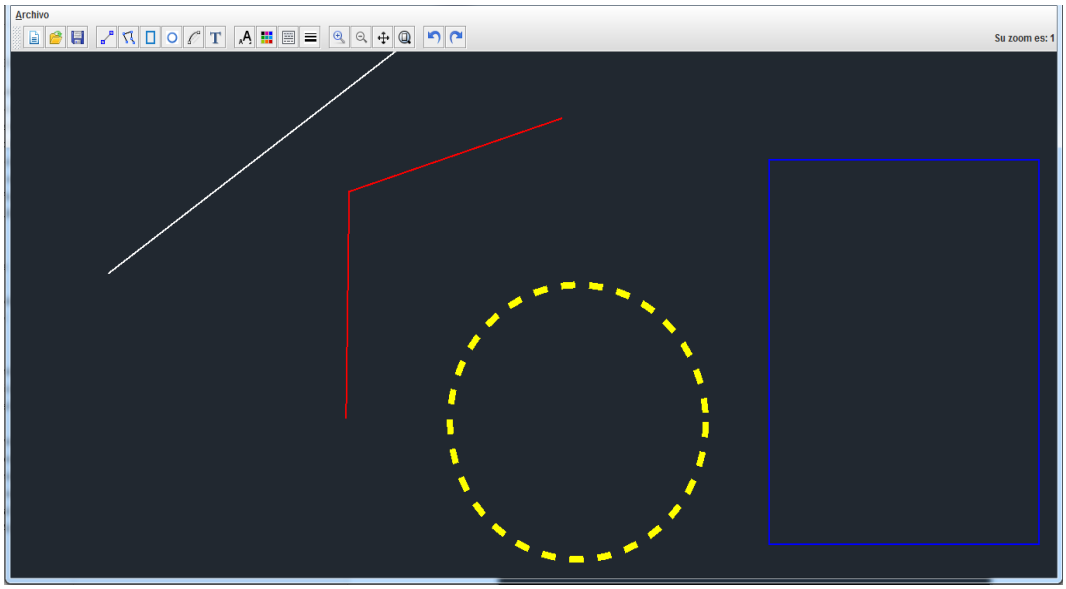

Ilustración 41: Ejemplo de dibujo con zoom 1 y desplazamiento con flechas derecha y abajo

Teniendo el espacio de diseño como muestra la *Ilustración 37*, si el usuario pulsa sobre el botón “Ajustar a la pantalla”  el espacio de diseño quedará como muestra la *Ilustración 34.* Este proceso no será instantáneo debido a que la aplicación tiene que realizar una serie de cálculos.

Por último, el usuario podrá eliminar las últimas entidades añadidas y podrá volver a añadir las eliminadas. Esta funcionalidad de “Deshacer” y “Rehacer” se aplica **SÓLO A LA ADICIÓN DE ENTIDADES**.

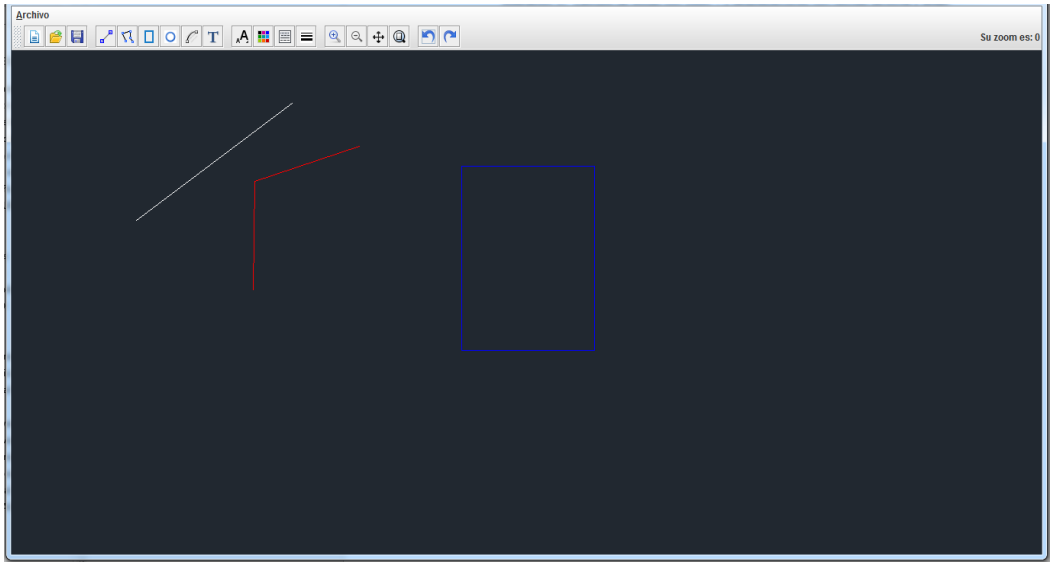

Ilustración 42: Espacio de diseño tras pulsar dos veces sobre la acción "Deshacer"

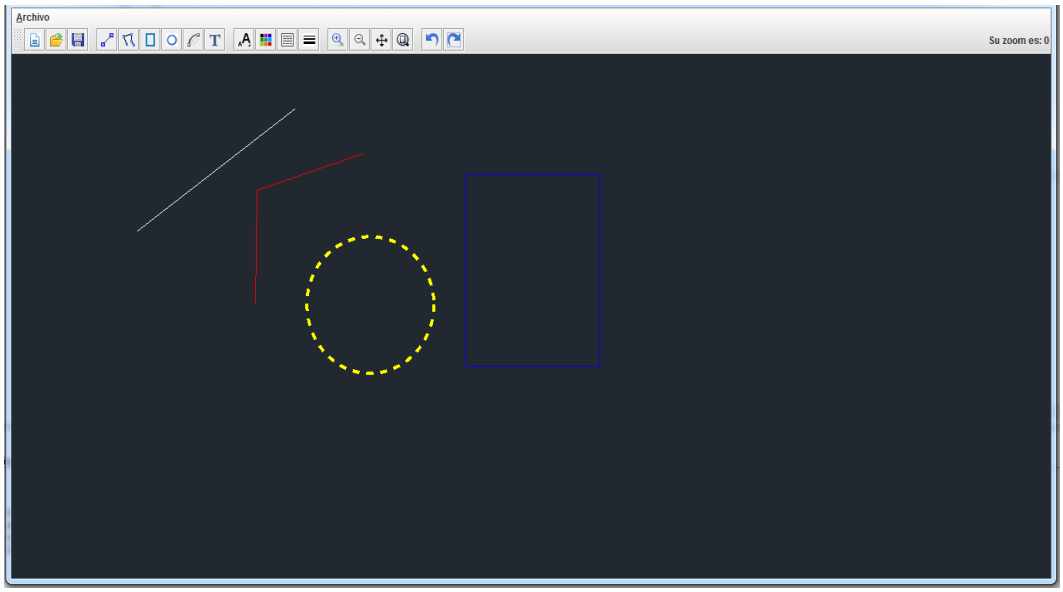

Ilustración 43: Espacio de diseño tras pulsar una vez sobre "Rehacer"

Para los ficheros, el usuario podrá Guardar el fichero que está editando mediante el proceso *Archivo -> Guardar* o bien pulsando sobre el botón de Guardar 

Sin embargo si quiere guardar el fichero con otro nombre deberá acceder al menú *Archivo* y seleccionar la opción *Guardar Como*. Para exportarlo a otros formatos, en la opción *Exportar* del menú *Archivo* se despliega un submenú en el que el usuario podrá elegir el formato correspondiente. Tras pulsar sobre él se abrirá una ventana del navegador donde el usuario puede elegir la ruta y el nombre con el que quiere guardar su fichero en el nuevo formato.

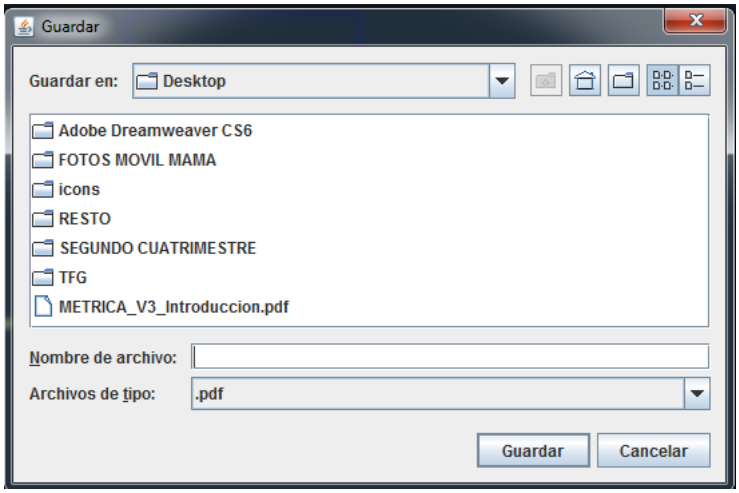

Ilustración 44: Ventana del navegador para guardar en formato PDF

Para cerrar la aplicación el usuario podrá elegir la opción *Archivo -> Salir* o pulsar sobre el botón situado en la esquina superior derecha de la ventana por defecto en los sistemas operativos *Windows*.